

Extracting recipe ingredients from cookbooks

by
Torsten Knauf

A thesis presented for the degree of
Master of Science



Research Group for Communication Systems
at
Faculty of Engineering
Christian-Albrechts-Universität zu Kiel
Germany
31.03.2017

Supervisor: Prof. Dr.-Ing. Norbert Luttenberger
Dr.-Ing. Jesper Zedlitz

Statutory Declaration

I declare that I have developed and written the enclosed Master Thesis completely by myself, and have not used sources or means without declaration in the text. Any thoughts from others or literal quotations are clearly marked. The Master Thesis was not used in the same or in a similar version to achieve an academic grading or is being published elsewhere.

Location, Date

Signature

webseite, url, git...

Problemstellung

Beitrag
zur
Forschung

Wie
diese
Arbeit zu
lesen ist

Ziel dieser Arbeit ist die Erstellung einer digitalen Edition des Buches *Praktisches Kochbuch für die gewöhnliche und feinere Küche* (Davidis, 1849), welche für kulinarische Analysen genutzt werden kann. Eine Transkription des Kochbuches wurde bereits vom Deutschen Textarchiv angefertigt. Ein beispielhafter Auszug ist in Abb. 1 zu sehen. Abb. 1a zeigt den Scan eines Rezeptes, Abb. 1b die textuelle Form und Abb. 1c eine bereits von uns überarbeitete Version der TEI-basierten Transkription.

Abb. 1: Transkription von Davidis's Kochbuch

Abb. 1a (Deutsches
Textarchiv (A), 2008)

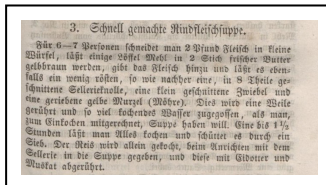


Abb. 1b (Deutsches
Textarchiv (A), 2008)

3. Schnell gemachte Rindfleischsuppe.
Für 6—7 Personen schneidet man 2 Pfund Fleisch in kleine Würfel, läßt einige Löffel Mehl in 2 Stuch frischer Butter gelbbraun werden, gibt das Fleisch hinzu und läßt es ebenfalls ein wenig rösten, so wie nachher eine, in 8 Theile geschnittene Sellerieknolle, eine klein geschnittene Zwiebel und eine geriebene gelbe Murrel (Möhre). Dies wird eine Weile gerührt und so viel kochendes Wasser zugegossen, als man, zum Einkochen mitgerechnet, Suppe haben will. Eine bis 1½ Stunden läßt man Alles kochen und schüttet es durch ein Sieb. Der Reis wird allein gekocht, beim Anrichten mit dem Sellerie in die Suppe gegeben, und diese mit Eidotter und Muskat abgerührt.

Abb. 1c

```
<cue:recipe type="Suppen." rcp-id="B-3">
<head>Schnell gemachte Rindfleischsuppe.</head>
<p>Für 6—7 Personen schneidet man 2 Pfund
Fleisch in kleine Würfel, läßt einige Löffel
Mehl in 2 Stuch frischer Butter gelbbraun
werden, gibt das Fleisch hinzu und läßt es
ebenfalls ein wenig rösten, so wie nachher
eine, in 8 Theile geschnittene
Sellerieknolle, eine klein geschnittene
Zwiebel und eine geriebene gelbe Murrel
(Möhre). Dies wird eine Weile gerührt und so
viel kochendes Wasser zugegossen, als man,
zum Einkochen mitgerechnet, Suppe haben will.
Eine bis 1½ Stunden läßt man Alles kochen und
schüttet es durch ein Sieb. Der Reis wird
allein gekocht, beim Anrichten mit dem
Sellerie in die Suppe gegeben, und diese mit
Eidotter und Muskat abgerührt. </p>
</cue:recipe>
```

Für kulinarische Analysen sind insbesondere Zutatenlisten wünschenswert, welche bei den Rezepten nicht vorhanden sind. Für eine maschinenlesbare Aufarbeitung wird ein **Domänen-spezifisches Vokabular** benötigt. Idealerweise können aus den mit dem Vokabular ausgezeichneten Rezepten anschließend die Zutatenlisten automatisch extrahiert werden. Da die manuelle Auszeichnung zeitaufwendig und fehleranfällig ist, forschen wir des Weiteren im Bereich **Information Extraction** an Möglichkeiten zum automatischen Auszeichnen. Beides sind keine trivialen Probleme.

Domänen-spezifisches Vokabular

Bestehende Vokabulare ermöglichen keine kulinarische Analyse. TEI hat kein spezifisches Vokabular um Rezepte auszuzeichnen. Schema.org/Recipe ist ein Vokabular, welches das Ziel hat, Treffer von Suchmaschinen anzureichern. Es wird davon ausgegangen, dass jedes Rezept eine Zutatenliste hat. In dieser wird jede Zeile beispielsweise mit Microdata angereichert; z. B. `2 EL Zucker`. Für den Computer stellt dieser Tag-Inhalt jedoch nur einen String dar. Für eine kulinarische Analyse müsste die Mengenangabe (2), die Einheit (EL), sowie die Zutat (Zucker) explizit ausgezeichnet sein. Das Auszeichnen von Zutaten im Fließtext, was wir mangels bestehender Zutatenliste machen müssen, muss darüber hinaus einige Sonderfälle beachten. Dies verdeutlichen folgende zwei Beispielsätzen aus dem Kochbuch:

1. „Der [Englische] Soja macht die Suppe gewürzreicher, kann jedoch gut wegbleiben, und statt Madeira kann man weißen Franzwein und etwas Rum nehmen.“ (Davidis,

1849, S. 33 f.)

2. „Das Kalbfleisch wie in No. 1, nach der Personenzahl, doch etwas reichlicher genommen, da solches weniger Kraft gibt, als Rindfleisch.“ (Davidis, 1849, S. 30)

In Erstens ist Soja eine optionale Zutat. Je nachdem ob sie verwendet wird, hat das Gericht eine unterschiedliche Geschmacksrichtung. In das Rezept gehören auch nicht Madeira, weißer Franzwein und Rum, sondern Madeira oder weißer Franzwein und Rum. Somit ist es nötig, bei der Auszeichnung zwischen **optionalen** und **alternativen Zutaten** zu unterscheiden. Des Weiteren sind alle drei Zutaten vage Begriffe. Rum gibt es zum Beispiel in vielen verschiedenen Preisklassen und mit unterschiedlichen Geschmacksausprägungen. Sofern vorhanden ist daher eine Präzisierung wünschenswert. Idealerweise wären **Zutaten bereits weltweit frei verfügbare und eindeutige Ressourcen mit abrufbaren Nährwertangaben**. Ein Mapping zwischen den verwendeten Zutaten pro Rezept und so einem Ressourcen-Bestand würde eine kulinarische Auswertung leicht und transparent machen. Leider gibt es unseren Wissens nach solche Ressourcen noch nicht.

In Zweitens ist Rindfleisch keine Zutat, sondern dient nur als Vergleich für eine ungefähre Mengenangabe. Wenn dieser Satz nur schnell gelesen wird, wird Rindfleisch hingegen leicht als Zutat eingeordnet (mir selber ist es erst beim dritten Mal lesen aufgefallen, dass es gar keine Zutat ist). Dies zeigt zum einen, dass beim Auszeichnen leicht Fehler gemacht werden können und zum anderen, dass nicht alle Zutaten als Zutaten des Rezepts auszuzeichnen sind.

Information Extraction in der Koch-Domäne

Information Extraction beschäftigt sich damit, nützliche Informationen aus unstrukturiertem Text zu extrahieren. Es wurde erstmals in (Feldman, Dagan, 1995) als eigene Disziplin erwähnt. Ein allgemeiner Überblick kann in (Hohto et al., 2005) gefunden werden. Die Algorithmen versuchen meist bestehende Semi-Strukturen auszunutzen. In der Koch-Domäne haben wir drei unterschiedliche automatisierte Ansätze gefunden, auf die wir genauer eingehen werden:

- Regular Expression-based
- Conditional Random Field-based
- Dictionary- and rule-based

Nachdem die Zutaten von einem Programm extrahiert wurden, müssen die extrahierten Entities zum Auszeichnen nur noch entsprechend dem Domänen-spezifischen Vokabular zurückgeschrieben werden.

Unten stehend ist zur Veranschaulichung eine interaktive Grafik. Diese zeigt alle nötigen Schritte und Zwischenergebnisse von der Transformation des gedruckten Buchs bis hin zur ausgezeichneten digitalen Version als Webseite. Als erstes muss das gedruckte Buch transkribiert werden. Anschließend entwickeln wir eine Auszeichnungssprache, die eine kulinarische Analyse ermöglicht. Danach beschäftigen wir uns mit Programmen, welche die Rezepte automatisch auszeichnen. Abschließend werden die ausgezeichneten Rezepte in eine Webseite umgewandelt. Aufbauend auf dieser kann eine kulinarische Analyse durchgeführt werden, was nicht mehr Teil dieser Informatik-Arbeit ist. Wenn in dem jeweiligen Arbeitsschritt eine Transformation stattgefunden hat, ist rechts die Grundlage zu sehen und links, die neue, transformierte Version.

Beitrag zur Forschung

Der Beitrag dieser Arbeit zu dem aktuellen Forschungsstand lässt sich wie folgt zusammenfassen:

- Die Feststellung, dass bestehende Auszeichnungssprachen nicht für eine kulinarische Analyse geeignet sind
- Entwicklung einer Auszeichnungssprache für kulinarische Analysen
- Die Erkenntnis, dass Information Extraction mittels Conditional Random Fields für unsere Problemstellung nicht zielführend ist
- Erstellung eines dictionary- and rule-based Prototypen
- Dokumentation sowie Durchführung aller nötigen Arbeitsschritte, wie ein gedrucktes Buch digital zu einer Webseite aufgearbeitet werden kann

Wie diese Arbeit zu lesen ist

Diese Webseite ist das Ergebnis meiner Master-Arbeit mit dem Titel *Extracting recipe ingredients from cookbooks*.

Abb. 2a zeigt die chronologische Reihenfolge, in welche diese Arbeit zu lesen ist. Die Problemstellung wird am Anfang dieser Seite erläutert. Unter dem Menüpunkt Digitale Edition sind die weiteren Abschnitte zu finden. Der erste Abschnitt behandelt aufbauend auf der Version des Deutschen Textarchivs unsere Transkription des gedruckten Buchs von Frau Davidis. Im zweiten Abschnitt stellen wir bestehende Auszeichnungssprachen in der Koch-Domäne vor. Da diese nicht für eine kulinarische Analyse geeignet sind, definieren wir für diesen Zweck im dritten Abschnitt die Auszeichnungssprache cueML. Der vierte Abschnitt beschäftigt sich mit der Information Extraction in der Koch-Domäne, mit dem Ziel später Rezepte automatisch mit cueML auszeichnen zu können. Darauf aufbauend werden im fünften Abschnitt zwei Prototypen zur Extraktion von Entities in der Koch-Domäne entwickelt. Die Evaluation dieser Prototypen ist Gegenstand des sechsten Abschnitts. Der siebte Abschnitt enthält einige Kommentare unsererseits zu dieser Arbeit. Der Menüpunkt Rezepte zeigt die von uns aufgearbeiteten und in HTML transformierten Rezepte Frau Davidis'. Der Menüpunkt Blog gibt Einblicke in unsere Arbeitsweise. Diese beiden Menüpunkte sind als bonus Material für den interessierten Leser zu verstehen.

Abb. 2b zeigt, wie der Aufbau der Arbeit inklusive der Unterabschnitte als klassisches Inhaltsverzeichnis aussieht.

Abb. 2a: Navigationsbar

Davidis' Kochbuch 1849	1. Problemstellung	Digitale Edition ▾	Bonus Material Rezepte Blog	Impressum
	Transkription	2.		
	Auszeichnen von Rezepten	3.		
	Domänen-spezifisches Vokabular cueML	4.		
	Information Extraction in der Koch-Domäne	5.		
	Automatisches Auszeichnen mit cueML	6.		
	Evaluierung	7.		
	Kommentare	8.		
	Zusammenfassung & Ausblick	9.		
	Quellen			


Abb. 2b: Klassisches Inhaltsverzeichnis

1	Transkription
2	Auszeichnen von Rezepten
2.1	Informationen zugänglich machen
2.2	TEI: Text Encoding Initiative
2.3	Schema.org/Recipe
2.4	In kommerziellen Kochseiten
3	Domänen-spezifisches Vokabular <i>cueML</i>
3.1	Anforderungen
3.2	<i>cueML</i>
4	Information Extraction in der Koch-Domäne
4.1	Regular Expression-based
4.2	Conditional Random Field-based
4.2.1	CRF erklärt
4.2.2	Implementierung der NYT
4.3	Dictionary- and rule-based
5	Automatisches Auszeichnen mit <i>cueML</i>
5.1	CRF-based Prototyp
5.2	Dictionary- and rule-based Prototyp
5.2.1	Version 0.1
5.2.2	Version 0.2
6	Evaluierung
6.1	Recall & Precision
6.2	CRF-based Prototyp
6.3	Dictionary- and rule-based Prototyp
6.3.1	Version 0.1
6.3.2	Version 0.2
7	Kommentare
7.1	Praktischer Nutzen der Arbeit
7.2	Schwierigkeiten für die kulinarische Analyse mit <i>cueML</i> und Frau Davidis' Kochbuch
7.3	Wissenschaftliches Arbeiten im 21. Jahrhundert
7.4	Wissen ist Macht

7.5 Final Retrospective
8 Zusammenfassung & Ausblick
8.1 Zusammenfassung
8.2 Ausblick
9 Quellen

Transkription

Das Deutsche Textarchiv (DTA) erfasst strukturelle und linguistische Merkmale deutschsprachiger Texte des 17. bis 19. Jahrhunderts, mit dem Ziel „ein ausgewogenes historisches Referenzkorpus in deutscher Sprache zu schaffen“ (Deutsches Textarchiv (B), 2017). Unter anderem stellen sie eine frei verfügbare Transkription von Davidis' Kochbuch in (Deutsches Textarchiv (A), 2008) zur Verfügung. Abb. 1a zeigt den Scan eines Rezeptes und Abb. 1b die in TEI transkribierte Version des Rezeptes. Wir sind allerdings nicht an linguistischen oder strukturellen Merkmalen interessiert, sondern bereiten eine kulinarische Analyse vor. Dementsprechend haben wir die Kodierung des DTA für unsere Zwecke überarbeitet. Dies ist in Abb. 1c zu sehen.

Abb. 1a (Deutsches Textarchiv (A), 2008)	Abb. 1: Transkription von Davidis' Kochbuch Abb. 1b (Deutsches Textarchiv (A), 2008)	Abb. 1c
	<pre><div n="3"> <head>3. Schnell gemachte Rindfleischsuppe.</head> <lb/> <p>Für 6-7 Personen schneidet man 2 Pfund Fleisch in kleine Würfel, läßt einige Löffel Mehl in 2 Stuch frischer Butter gelbbraun werden, gibt das Fleisch hinzu und läßt es eben- falls ein wenig rösten, so wie nachher eine, in 8 Theile ge- schnittene Sellerieknolle, eine klein geschnittene Zwiebel und eine geriebene gelbe Murrel (Möhre). Dies wird eine Weile gerührt und so viel kochendes Wasser zugegossen, als man, zum Einkochen mitgerechnet, Suppe haben will. Eine bis 1½ Stunden läßt man Alles kochen, beim Anrichten mit dem Selterie in die Suppe gegeben, und diese mit Eidotter und Muskat abgerührt.</p> </div></pre>	<pre><cue:recipe type="Suppen." rcp-id="B-3"> <head>Schnell gemachte Rindfleischsuppe.</head> <p>Für 6-7 Personen schneidet man 2 Pfund Fleisch in kleine Würfel, läßt einige Löffel Mehl in 2 Stuch frischer Butter gelbbraun werden, gibt das Fleisch hinzu und läßt es ebenfalls ein wenig rösten, so wie nachher eine, in 8 Theile geschnittene Sellerieknolle, eine klein geschnittene Zwiebel und eine geriebene gelbe Murrel (Möhre). Dies wird eine Weile gerührt und so viel kochendes Wasser zugegossen, als man, zum Einkochen mitgerechnet, Suppe haben will. Eine bis 1½ Stunden läßt man Alles kochen und schüttet es durch ein Sieb. Der Reis wird allein gekocht, beim Anrichten mit dem Selterie in die Suppe gegeben, und diese mit Eidotter und Muskat abgerührt. </p> </cue:recipe></pre>

Zum einen haben wir kodierte Sonderzeichen zu Gunsten der Lesbarkeit durch ihre entsprechende Standard-Pendants ersetzt; z.B. das „f“ (lange S) durch ein heute übliches „s“ (rundes S). Dazu haben wir strukturelle Merkmale wie die Kodierung von Zeilenumbrüchen (<lb/>) oder die Verschachtelungstiefe innerhalb des Buches (<div n="3"/>) entfernt. Stattdessen haben wir jedes Rezept, welches zu unseren Zielobjekten gehört, in einem einheitlichen Tag (<cue:recipe>) gekapselt. Zusätzlich haben wir das Tag für Rezepte mit der Kapitelüberschrift als Typ angereichert (type="Suppen"), sowie eine eindeutige ID vergeben (rcp-id="B-3"). Des Weiteren haben wir die interne Kapitelnummerierung (3.) aus der Rezept-Überschrift entfernt, da diese nichts zur kulinarischen Analyse beiträgt, jedoch bei separater Betrachtung irreführend ist.

Der Vollständigkeit halber sei erwähnt, dass das DTA die Transkription von Nicht-Muttersprachlern im Double-Keying-Verfahren hat durchführen lassen. Ihre Intention dabei ist, dass so unbewusste Modernisierungen, Korrekturen und Wertungen beim Transkribieren vermieden werden. Allerdings führt das zu Fehlern wie *Eßlöffel* statt *Eßlöffel*. Dem aufmerksamen Leser wird aufgefallen sein, dass Ersteres (*Eßlössel*) kein deutsches Wort ist. In diesem Beispiel wurde fälschlicherweise das „ff“ für zwei lange „s“ gehalten. Sofern uns solche Fehler aufgefallen sind, haben wir sie in unserer Transkription korrigiert.

Abschließend sei darauf hingewiesen, dass sämtliche Anerkennung der Arbeit, die Kodierung des DTA zu unserer Kodierung zu transformieren, meinem betreuenden Prof. Herrn Luttenberger gebührt, wie in diesem Blog-Abschnitt zu lesen ist.

Auszeichnen von Rezepten

Informationen
zugänglich
machen

TEI: Text
Encoding
Initiative

Schema.org/Recipe

In
kommerziellen
Kochseiten

Nachdem wir im vorherigen Abschnitt eine Transkription zu Davids' Kochbuch erarbeitet haben, wollen wir uns nun damit beschäftigen, wie wir die Informationen aus dem Buch automatisch auswerten können. Dafür gehen wir zuerst darauf ein, wie Informationen zugänglich gemacht werden können. Anschließend betrachten wir existierende Auszeichnungssprachen, die Informationen aus unserer Domäne zugänglich machen.

Informationen zugänglich machen

"The key to utilizing the knowledge of an application domain is identifying the basic vocabulary consisting of terms or concepts of interest to a typical user in the application domain and the interrelationships among the concepts in the ontology." (Kashyap et al., 2008, S. 30 f.)

Eine *Ontology* wiederum ist ein „set of interest in a particular information domain and the relationships among them“ (Kashyap et al., 2008, S. 79). Ein wohldefiniertes Vokabular kann demnach eine Ontology sein. Dieses kann beispielsweise durch ein *Extensible Markup Language Schema (XML Schema)* definiert werden (vgl. Kashyap et al., 2008, S. 84 ff.). Alle Informationen eines XML Dokumentes nach einem Schema können dann automatisch extrahiert werden.

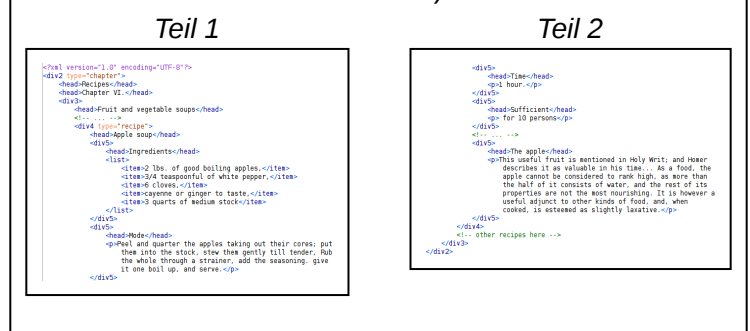
Das Ziel ist somit, die Rezepte aus Davidis' Kochbuch in einer Auszeichnungssprache zu erfassen, dessen Vokabular und Beziehungen untereinander wohldefiniert sind und die den Bereich einer kulinarischen Analyse abdeckt. Das Erfassen von Informationen durch eine Auszeichnungssprache nennen wir *auszeichnen*. Folgend stellen wir einige Auszeichnungssprachen für unsere Koch-Domäne vor.

TEI: Text Encoding Initiative

TEI ist ein Standard, um Texte digital darzustellen, mit dem Schwerpunkt die Texte für Geistes- und Sprachwissenschaften maschinenlesbar zu machen. (TEI-Konsortium (A), 2017)

Wie in Abb. 1 zu sehen ist, wird die Struktur eines Kochbuches eins zu eins abgebildet. Die strukturelle Verschachtelungstiefe wird

Abb. 1: Rezept mit TEI (Nach: TEI-Konsortium (B), 2017, S. 1036 f.)



durch die nummerierten *div*-Elemente abgebildet, die Überschriften durch *head*-Elemente, die Paragraphen in *p*-Elementen und die Listen durch *list*- und *item*-Elemente.

Schema.org/Recipe

Schema.org/Recipe ist ein Vokabular, um Treffer von Suchmaschinen anzureichern. Die einzelnen Bereiche eines Rezeptes werden durch Microdata, RDFa oder JSON-LD kenntlich gemacht und können so von einer Suchmaschine ausgelesen werden. Abb. 2 zeigt einen Suchtreffer von Google zur Anfrage „Pfannekuchen“.

Abb. 2: Beispielhafter Google Treffer zu „Pfannekuchen“



In Abb. 3 sind die hinterlegten Meta-Informationen auf der Webseite des Treffers in JSON-LD-Format dargestellt. Es wird unter anderem ersichtlich, dass das Vorschau-Bild in Google eine Vorschau des *image*-Attributes ist. Die Bewertung ist der Wert des *aggregateRating.ratingValue*-Attributes und der Vorschau-Text ist der Beginn des *description*-Attributes.

Die wesentlichen Bestandteile der Webseite mit dem eigentlichen Rezept sind **hier** zu sehen.

Abb. 3: JSON-LD Anwendung von Schema.org/Recipe auf das Pfannekuchen-Rezept (Nach: 06onkel von Chefkoch.de, 2017)

```
<script type="application/ld+json">
{
  "@context": "http://schema.org",
  "@type": "Recipe",
  "datePublished": "2005-07-20",
  "description": "Pfannekuchen, ein gutes Rezept [...]",
  "image": "http://i...jpg",
  "recipeIngredient": [
    "250 g Mehl", "4 Ei(er)", "Salz", "300 ml Milch",
    "300 ml Mineralwasser, (mit Kohlensäure)", "Öl"
  ],
  "name": "Pfannekuchen",
  "author": {
    "@type": "Person",
    "name": "06onkel"
  },
  "prepTime": "PT30M",
  "recipeInstructions": "Den ganzen Teig mit [...]",
  "recipeYield": "4",
  "aggregateRating": {
    "@type": "AggregateRating",
    "ratingValue": "4.3",
    "reviewCount": "93",
    "worstRating": 0,
    "bestRating": 5
  },
  "recipeCategory": ["Braten", [...]]
}
</script>
```

In kommerziellen Kochseiten

Wie im vorherigen Abschnitt gesehen, verwendet Chefkoch.de Schema.org/Recipe. Zusätzlich speichert Chefkoch.de jedes Rezept in einem JSON-Objekt, was am Beispiel des Pfannekuchen-Rezeptes in Abb. 4 zu sehen ist. Neben einer zusätzlichen ID (id) für das Rezept, werden die Einträge der Zutatenliste, welche bereits mit Schema.org/Recipe erfasst wurden, genauer aufgespalten. Jede Zutat hat eine eigene ID (id), Mengenangabe (base_amount) mit Mengeneinheit (amount_unit), sowie optional einen Kommentar (suffix).

Abb. 4: Extra JSON-Attribute für Zutaten (Nach: 06onkel von Chefkoch.de, 2017)

```
<script type="text/javascript">
var _simplora_params = { recipes: [] };
_simplora_params.recipes.push({
  id: '363861121870267',
  name: 'Pfannekuchen',
  servings: 4,
  ingredients: [
    { id: '71', base_amount: '62.5', amount_unit: 'g', name: 'Mehl' },
    { id: '36', base_amount: '1', amount_unit: '', name: 'Ei(er)' },
    { id: '6', base_amount: '0', amount_unit: '', name: 'Salz' },
    { id: '73', base_amount: '75', amount_unit: 'ml', name: 'Milch' },
    { id: '1012', base_amount: '75', amount_unit: 'ml', name: 'Mineralwasser', suffix: ', (mit Kohlensäure)' },
    { id: '184', base_amount: '0', amount_unit: '', name: 'Öl' },
  ]
});
</script>
```

Cooking.nytimes.com verwendet ebenfalls Schema.org/Recipe. Aus (Greene, 2015) geht hervor, dass Cooking.nytimes.com zusätzlich ähnlich wie Chefkoch.de die Zutaten der Rezepte in einer Datenbank abspeichert, aufgespalten nach Name, Mengenangabe, Mengeneinheit, Kommentar und Anderes.

Domänen-spezifisches Vokabular *cueML*

Anforderungen

cueML

Zunächst werden hier die Anforderungen an ein Vokabular in der Koch-Domäne dargelegt, welche sich aus unserem Kochbuch ergeben. Da die zuvor vorgestellten Auszeichnungssprachen diesen Anforderungen nicht genügen, stellen wir anschließend unser selbst entwickeltes *cueML*-Vokabular vor.

Anforderungen

Folgend erklären wir, was für uns ein Rezept ausmacht und was das Vokabular dementsprechend erfassen können muss. Die einzelnen Punkte werden zusammenfassend am Ende noch einmal aufgelistet.

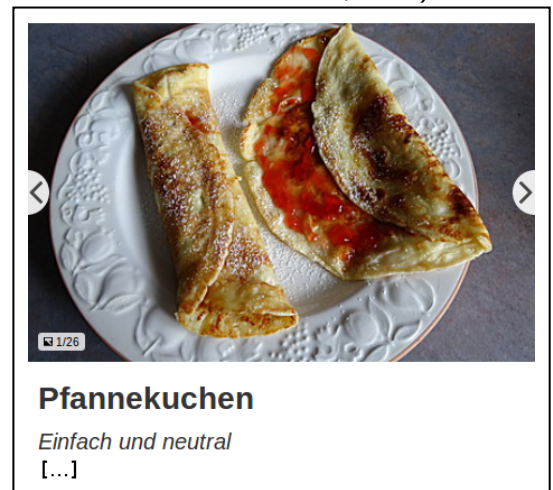
Ein Rezept besteht für uns nicht nur aus reinem Text. Aus unserer Sicht **ist ein Rezept in Themenbereiche unterteilt**.

Das Beispiel-Rezept aus Abb. 1 ist eindeutig in einen Bereich für ein Bild, eine Überschrift/Namen einen einleitenden Text, usw. aufgeteilt. Ein Rezept hat für uns im Allgemeinen folgende klar strukturierte und voneinander separierte Bereiche:

- Den Namen als Überschrift
- Optional einen einleitenden, möglichst ansprechenden Text wie zum Beispiel „Meine absoluten Lieblings-Pfannekuchen nach dem geheimen Rezept meiner Oma!“ (welches wohl spätestens nach der Veröffentlichung des Rezeptes nicht mehr ganz so geheim ist)
- Optional und sehr empfehlenswert ein Bild vom Gericht
- Geschätzte Zubereitungszeit
- Nährwertangaben pro Person
- Anzahl der produzierten Portionen
- Eine Zutatenliste
- Zubereitungs-Anweisungen

Ein Vokabular sollte diese Struktur erfassen können. Sofern in einem Rezept die Struktur noch nicht gegeben ist, sollten die im Rezept vorhandenen Informationen

Abb. 1: Beispiel Rezept (Nach: 06onkel von Chefkoch.de, 2017)



nach einer Auszeichnung mit dem Vokabular automatisch in so eine Struktur umgewandelt werden können.

In den Rezepten aus unserem Kochbuch ist keine Zutatenliste vorhanden. Die Zutaten sind allerdings in den Zubereitungs-Anweisungen zu finden. Dementsprechend sollte nach einer Auszeichnung mit dem Vokabular **eine Zutatenliste aus den Zubereitungs-Anweisungen eines Rezeptes extrahierbar sein**. Daraus ergeben sich weitere Anforderungen, wie folgende zwei Beispiel-Sätze verdeutlichen:

1. „*Der [Englische] Soja macht die Suppe gewürzreicher, kann jedoch gut wegbleiben, und statt Madeira kann man weißen Franzwein und etwas Rum nehmen.*“ (Davidis, 1849, S. 33 f.)
2. „*Man kocht nach No. 1 eine gute Bouillon [...]*“ (Davidis, 1849, S. 32)

In Erstens *kann* Soja zum würzen verwendet werden. Da Soja jedoch sehr geschmacksintensiv ist, hat das Gericht, je nachdem ob Soja verwendet wird, eine ganz andere Geschmacksrichtung. Dementsprechend wäre eine Festlegung, ob Soja in die Zutatenliste kommt oder nicht, eine Reduzierung des Rezeptes. Daher soll das Vokabular **optionale Zutaten** von Obligatorischen unterscheiden können.

Des Weiteren gehören in das Rezept nicht Madeira, weißer Franzwein *und* Rum, sondern Madeira *oder* weißer Franzwein und Rum. Also muss das Vokabular **alternative Zutaten** abbilden können.

Erstens enthält somit genau genommen vier mögliche, unterschiedliche Rezepte. Je zwei ob die Soja verwendet wird oder nicht, sowie je zwei ob Madeira oder weißer Franzwein und Rum verwendet wird. Nach der Auszeichnung sollen daher aus den Rezepten alle möglichen Rezepte abgeleitet werden können.

In Zweitens soll *nach No. 1* eine Bouillon gekocht werden. Rezept No. 1 ist jedoch nicht ein Rezept für Bouillon, sondern ein Rezept für Rindfleischsuppe, in welches das Rezept für Bouillon integriert ist (Davidis, 1849, S. 28). Dementsprechend soll das Vokabular **Verweise auf (Teil-) Rezepte** ermöglichen.

Für eine kulinarische Analyse sind offensichtlich nicht nur die Zutaten wichtig, sondern auch die Mengenangaben der Zutat, wobei eine Mengenangabe ohne Einheit wertlos ist. Daher wird im Vokabular zusätzlich eine Möglichkeit zur Auszeichnung von **Mengenangaben** und **Mengeneinheiten** benötigt.

Im Sinne des semantischen Webs ist es zudem wünschenswert, dass die **Zutaten und Mengeneinheiten als weltweit frei verfügbare und eindeutige Ressourcen** (mittels URIs) verstanden werden. Eine gute und anschauliche Erklärung des Begriffes *semantisches Web* ist in (Berners-Lee et al., 2001) zu finden. Zu der URI einer Zutat können Nährwertangaben und Einordnungen in Taxonomien wie „ist-vegetarisch“ hinterlegt werden. Die Mengeneinheiten sind erst dann sinnvoll verwendbar, wenn Informationen zur Umrechnung in standardisierte Einheiten vorliegen; z. B. die Mengenangabe „für 8 Pfennig [...] Weißbrot“ (Davidis, 1849, S. 35 f.) ist ohne die Information, wie viel Gramm/Scheiben Weißbrot man nach Meinung von Frau Davidis für 8 Pfennig kriegte, wertlos. Erst die Verknüpfung beider Ressourcen ermöglicht eine transparente Extraktion von Nährwertangaben aus einer Zutatenliste.

Zusammengefasst ergeben sich daraus für uns folgende Anforderungen:

- Auszeichnung von Themenbereichen
- Unterscheidung von obligatorischen Zutaten, optionalen Zutaten und alternativen Zutaten
- Extraktion einer Zutatenliste aus den Zubereitungs-Anweisungen
- Verweise zwischen Rezepten
- Zutaten und Mengeneinheiten als frei verfügbare und eindeutige Ressourcen

cueML

Die zuvor vorgestellten Auszeichnungssprachen erfüllen die oben genannten Anforderungen an das benötigte Vokabular nicht, wie folgend knapp an je einem Punkt erläutert wird: *TEI* erhält zwar die Struktur des Rezeptes, stellt allerdings keine Vokabeln zur Verfügung, um die einzelnen Themenbereiche wie Zutatenliste oder Zubereitungs-Anweisung als solche auszuzeichnen. *Schema.org/Recipe* bietet zwar die Möglichkeit Zutaten als solche zu markieren, aus dem eingeschlossenen Text der Markierungen kann der Computer jedoch weder die konkrete Zutat noch die Mengenangabe mit dazugehöriger Einheit ableiten. Die eigenen Vokabulare von *Chefkoch.de* und *Cooking.nytimes.com* zeichnen die bestehenden Zutatenlisten aus, welche bei unserem Kochbuch nicht vorhanden sind. Des Weiteren sind ihre Vokabulare nicht frei verfügbar und somit von uns sowieso nicht verwendbar.

Daher haben wir **culinary editions Markup Language (cueML)** entwickelt, was netterweise wie Kümmel auszusprechen ist. Eine beispielhafte Auszeichnung von einem Rezept aus unserem Kochbuch mit cueML ist in Abb. 2 zu sehen.

Abb. 2a: Unsere Transkription eines
Rezeptes aus dem Kochbuch

```
<cue:recipe type="Suppen." rcp-id="B-16">
<head>Mock Turtle Suppe.</head>

<p>Es wird hierzu für 24-30 Personen eine kräftige Bouillon von 8-10 Pfund Rindfleisch mit Wurzelwerk gekocht. Zugleich bringt man einen großen Kalbskopf, eine Schweineschnauze und Ohren, einen Ochsenaugen und eine geraucherte Ochsenzunge zu Feuer und kocht dies Alles gahr, aber nicht zu weich. Kalt, schneidet man es in kleine, länglich viereckige Stückchen, gibt das Fleisch in die Bouillon, nebst braunen Gewürz, ein Paar Messerspitzen Cayenne-Pfeffer, einige Kalbsnider in Stückchen geschnitten (siehe Vorbereitungsregeln), kleine Saucissen, so viel Kalbskopfbrühe, daß man hinreichend Suppe hat, und macht dies mit in Butter braun gemachtem Mehl gebunden. Nachdem dies Alles ½ Stunde gekocht hat, kommen noch Klöße von Kalbfleisch, einige hart gekochte Eier in Würfel geschnitten, ein Paar Eßlöffel Engl. Soja hinzu, und wenn die Klöße einige Minuten gekocht haben, ¼ Flasche Madeira und auch austern, wenn man sie haben kann. Dann wird die Suppe sogleich angerichtet. </p>

<note>Anmerk. Der Soja macht die Suppe gewürzreicher, kann jedoch gut wegleiben, und statt Madeira kann man weißen Franzwein und etwas Rum nehmen. Sowohl die Bouillon als Kalbskopf können schon am vorhergehenden Tage, ohne Nachtheil der Suppe, gekocht werden. </note>
</cue:recipe>
```

Abb. 2b: Gleiches Rezept mit cueML
ausgezeichnet

```
<cue:recipe type="Suppen." rcp-id="B-16">
<head>Mock Turtle Suppe.</head>

<p>Es wird hierzu für <cue:recipeField atleast="24" atMost="30" unit="people">24-30 Personen</cue:recipeField> eine kräftige <cue:recipeIngredient target="#Bouillon">Bouillon</cue:recipeIngredient> von 8-10 Pfund <cue:recipeIngredient ref="#Rindkuchfleisch" atleast="8" atMost="10" unit="Pfund">Rindfleisch</cue:recipeIngredient> mit <cue:recipeIngredient ref="#Wurzelwerk">Wurzelwerk</cue:recipeIngredient> gekocht. Zugleich bringt man einen großen <cue:recipeIngredient ref="#Kalbskopf" quantity="1">Kalbskopf</cue:recipeIngredient>, [...]</p>

<note>Anmerk. Der <cue:recipeIngredient ref="#Englische Soja" optional="True">Soja</cue:recipeIngredient> macht die Suppe gewürzreicher, kann jedoch gut wegleiben, und statt <cue:recipeIngredient ref="#Madeira" altGrp="1">Madeira</cue:recipeIngredient> kann man <cue:recipeIngredient ref="#weißer Franzwein" altGrp="2">weißen Franzwein</cue:recipeIngredient> und etwas <cue:recipeIngredient ref="#Rum" altGrp="2" quantity="etwas">Rum</cue:recipeIngredient> nehmen<cue:alt target="1 2"/>. [...]</note>
</cue:recipe>
```

Es kombiniert und erweitert *TEI* und *Schema.org/Recipe*. Da die Transkription bereits in *TEI* vorliegt, ist es naheliegend *TEI* zu erweitern. Das Vokabular von *Schema.org/Recipe* übernehmen wir. So kann aus cueML für Suchmaschinen leicht zusätzlich eine Auszeichnung mit *Schema.org/Recipe* abgeleitet werden. Zusätzlich erweitern wir es um Attribute für Mengenangaben (*quantity*, bzw. *atLeast* und *atMost*) und Mengeneinheiten (*unit*), sowie Möglichkeiten um optionale (*optional="True"*) und alternative Zutaten (*altGrp="id"* und *alt="id1 id2 [...]"*) auszuzeichnen. Des Weiteren führen wir Möglichkeiten ein, um eine Zutat auf Bereiche des Kochbuches verweisen zu lassen (*target="#id"*) sowie allgemeine Verweise innerhalb eines Rezeptes kenntlich zu machen (ein in Abb. 2b nicht vorhandenes *ref*-Element).

Einen Ressourcenbestand von Zutaten, wie in den Anforderungen beschrieben, konnten wir leider nicht finden. Daher

behelfen wir uns mit dem Bundeslebensmittelschlüssel (BLS). Der BLS enthält zu knapp 15.000 Zutaten und Gerichten 38 Nährwert-Informationen wie z. B. Ballaststoffe pro 100g. An sich ist er nicht frei verfügbar, wir dürfen jedoch mit ihm im Rahmen einer akademischen Lizenz arbeiten.

Abb. 3: Anbindung an den BLS

```
<cue:ingredient xal:id="Cayennepeffer" BLSref="R252000">
  <cue:prefBasicForm>Cayennepeffer</cue:prefBasicForm>
  <cue:altBasicForm>Cayenne-Pfeffer</cue:altBasicForm>
</cue:ingredient>
<cue:ingredient xal:id="Mlleder" BLSref="V562100">
  <cue:prefBasicForm>Mlleder</cue:prefBasicForm>
  <cue:altBasicForm>Kalbsmlleder</cue:altBasicForm>
  <cue:altBasicForm>Bries</cue:altBasicForm>
  <cue:altBasicForm>Kalbsmilch</cue:altBasicForm>
  <cue:note>Kalbsmlleder ist auch unter den Synonymen: Bries, Kalbsmilch bekannt. Kalbsmilch ist die Thymusdrüse des Kalbes. 100 g frische Kalbsmilch enthalten 99,8 kcal, 3,4 g Fett, 17,2 g Eiweiß, 77,8 g Wasser, 1,91 mg Eisen, 268 mg Cholesterin und 0,42 g Purine. Dieses Organ ist bei Jungtieren voll entwickelt. Bei erwachsenen Tieren bildet sich das Organ zurück. Kalbsmilch wird zur Herstellung von Spezialitäten, wie Suppen, Klöße und Ragout, verwendet. (http://www.cosmiq.de/qa/show/70827/was-ist-kalbsmlleder/)</cue:note>
</cue:ingredient>
<cue:ingredient xal:id="Saucisse" BLSref="W0000000">
  <cue:prefBasicForm>Saucisse</cue:prefBasicForm>
  <cue:note>Franz. Name für bestimmte Würste (s. https://fr.wikipedia.org/wiki/Saucisse)</cue:note>
</cue:ingredient>
<cue:ingredient xal:id="Butter" BLSref="Q610000">
  <cue:prefBasicForm>Butter</cue:prefBasicForm>
</cue:ingredient>
```

Abb. 3 zeigt, wie wir den BLS in cueML integriert haben. Jede Zutat wird auf ein *ingredient*-Element abgebildet und das *BLSref*-Attribut ist ein Verweis auf den eindeutigen BLS-Schlüssel. Zusätzlich geben wir eine Liste von möglichen Lemmata für die Zutaten sowie optional ein erläuterndes *note*-Element an.

Einen Ressourcenbestand für Mengenangaben konnten wir ebenfalls nicht finden. Auf einen Ressourcenbestand, der Frau Davidis' Einheiten wie *1 Maß*, oder *für 8 Pfennig Weißbrod* umrechnet, wird nicht weiter eingegangen, da das nicht Teil dieser Informatik-Arbeit ist. Unabhängig davon sei erwähnt, dass es ganz im Sinne der Programmier-Prinzipien *Separation of Concerns* und *DRY* keine gute Idee ist, die Umrechnung in der Auszeichnung vorzunehmen. Sollte sich beispielsweise bei späteren Recherchen ergeben, dass 1 Maß nach Frau Davids doch nicht 1l sondern 0,8 oder doch 1,2l entsprechen, müssten sämtliche ausgezeichnete Umrechnungen erneut vorgenommen werden. Bei einem Verweis auf eine URI, welche zu der Mengeneinheit 1 Maß von Frau Davidis gehört, muss nur der Eintrag bei der entsprechenden URI geändert werden. Daher übernehmen wir Frau Davidis' originalen Mengeneinheiten.

Im Gegensatz zu Schema.org/Recipe haben wir **cueML durch eine RELAX NG-Grammatik wohldefiniert**. Schema.org hat die Prämisse „some data is better than none“ (Schema.org, 2017) und validiert daher nicht gegen eine Grammatik. Für Suchmaschinen, die möglichst viele Daten erfassen wollen und die Zielgruppe der Schema.org-Vokabulare sind, ist das eine vertretbare Prämisse. Unsere Arbeitsgruppe ist dagegen der Überzeugung, dass die Validierung gegen eine Grammatik nicht schwer ist und einfache Fehler, wie beispielsweise Tippfehler verhindert. Darüber hinaus kann eine Grammatik gut als Dokumentation verwendet werden. Des Weiteren verstärkt sie die Idee des Vokabulares als Ontology. basisCueML.rng definiert die Kombination von TEI und Schema.org/Recipe und wurde von Prof. Dr.-Ing. Luttenberger mittels Roma erstellt. cueML_v05.rng definiert die Erweiterungen unseres Vokabulares und inkludiert basisCueML.rng.

Information Extraction in der Koch-Domäne

Regular
Expression-
based

Conditional
Random
Field-
based ▼

Dictionary-
and rule-
based

Folgend werden Algorithmen zum Extrahieren von Zutaten aus semi-strukturierten Text vorgestellt, die wir im Kontext der Koch-Domäne gefunden haben. Sind die Zutaten erst einmal von einem Programm extrahiert, kann das Programm diese leicht mit cueML auszeichnen und zurückschreiben. Das automatische Auszeichnen ist wünschenswert, **da das manuelle Auszeichnen...**

- **zeitaufwendig ist.** Die Erfahrung hat gezeigt, dass ich selber im Schnitt ca. 5 Minuten pro Rezept zum Auszeichnen brauche. Eine SHK, welche nicht im Umgang mit XML geübt war, hat freundlicherweise ebenfalls ein paar Rezepte ausgezeichnet. Sie hat im Schnitt sogar 15 Minuten für ein Rezept gebraucht.
- **fehleranfällig ist.** Diese Feststellung bei mir selbst, wie auch bei der SHK werden u.A. durch (Erdmann et al., 2001) und (Greene, 2015) bestätigt.
- **Domänen-spezifisches Wissen benötigt.** Wir sind z. B. über die Zutat *Scorzonerwurzel* gestolpert. Das Nachfragen bei einer Ernährungswissenschaftlerin der Kieler Uni ergab, dass damit wahrscheinlich Scorzone, bzw. Sommer-Trüffel gemeint ist. Später ist uns jedoch aufgefallen, dass Frau Davidis in viele Suppen Scorzonerwurzeln hineingibt. Dies hat bei uns die Frage aufgeworfen, ob sie wirklich oft empfiehlt, teure Sommer-Trüffel in Suppen zu zerkochen. Nach einer weiteren Recherche glauben wir nun, dass sie mit Scorzonerwurzel *Scorzonera hispanica* bzw. Schwarzwurzel meint, welche sicherlich in einer Suppe angebracht sind als Sommer-Trüffel.

Regular Expression-based

In (Skip The Pizza auf WordPress.org, 2012) kombiniert der Autor seine zwei Hobbies Programmieren und Kochen. Um Fragen beantworten zu können, wie *aus wie vielen Zutaten besteht ein Rezept im Schnitt?* oder *welche sind die meist benutzen Zutaten?*, will er die Zutaten aus über 29.000 Rezepten von Recipes wikia extrahieren.

Abb. 1 zeigt, wie die Rezepte in Recipes wikia abgespeichert sind. Offenbar haben die Rezepte die Semi-Struktur, dass jede Zutat innerhalb `[[...]]` eingeschlossen ist. Der Autor nutzt diese Semi-Struktur aus, um mittels Regulärer Ausdrücke die Zutaten zu extrahieren.

Abb. 1: Interne Struktur der Rezepte von Recipes wikia

```
* Makes 6 to 8 servings

== Ingredients ==
* 2 tbsp extra virgin [[olive oil]]
* 3 cloves [[garlic]], finely chopped
* 1½ pounds fresh ripe [[tomato]]es, seeded and chopped (about 3 cups)
* 1 tbsp [[tomato paste]]
* 1 tsp dried [[oregano]]
* ½ tsp [[ground red pepper]]
* ½ cup pitted brine cured [[black olives]] coarsely chopped
* 2 tbsp [[capers]]
* [[salt]] and [[pepper]]
* 1 pkg. thin [[spaghetti]] (16 oz)
* grated [[Parmesan cheese]]

== Directions ==
[...]
```

Conditional Random Field-based

Die New York Times (NYT) betreibt eine eigene Kochseite. Greene behauptet, dass sie mittels Linear-chain Conditional Random Fields (CRF) in der Lage sind „*automatisch den unstrukturierten Text von Rezepten in strukturierte Daten umzuwandeln*“ (Greene, 2015). Dies beinhaltet insbesondere das Extrahieren der Zutaten mit ihren Mengenangaben und Einheiten. Da diese Behauptung genau unserer Problemstellung entspricht, stellen wir zunächst die Idee von CRF vor und gehen anschließend auf die Implementierung von Greene ein.

CRF erklärt

CRF *orakelt* zu einem Vektor von Wörtern ein Vektor von Labels. Im Folgenden bezeichnen wir daher CRF auch als ein Orakel. Wenn wir dem Orakel z. B. den Vektor *[Man, süße, mit, 2, EL, Zucker, .]* geben, orakelt es uns idealerweise die Labels *[OTHER, OTHER, OTHER, QUANTITY, UNIT, INGREDIENT, OTHER]*. Wenn wir so ein Orakel hätten, würden wir unsere Rezepte in solche Wort-Vektoren aufspalten und dem Orakel übergeben. Aus der Antwort des Orakels könnten wir dann sofort die Zutaten mit Mengenangaben und Einheiten ablesen.

Jetzt stellt sich nur noch die Frage, woher man so ein Orakel bekommt. Folgend erläutern wir die Idee, wie so ein Orakel antrainiert werden kann. Eine detaillierte Einführung ist in (Sutton, McCallum, 2012) zu finden. Wir fangen damit an, dass wir dem Orakel eine Reihe von Wort-Vektoren geben und ihm für diese bereits die passenden Antworten von Label-Vektoren mitgeben. Solch eine Menge von Wort- und zugehörigen Label-Vektoren werden auch *Trainingsdaten* genannt. Fragen wir es nun nach den Labels von einen dieser Wort-Vektoren, könnte es uns einfach den zugehörigen Label-Vektor orakeln.

Spannend ist die Frage, wie das Orakel Labels für Wort-Vektoren rät, von denen es die Lösung nicht kennt. Dafür leitet es zunächst aus den Trainingsdaten eine multivariate Wahrscheinlichkeitsverteilung $p(X, Y)$ ab, die angibt, wie wahrscheinlich zu einem Wort-Vektor X ein Label-Vektor Y gehört. Das Orakel geht dabei von dem *vereinfachten Modell* aus, dass jedes Label $y_i \in Y$ nur vom vorherigen Label $y_{i-1} \in Y$ und dem entsprechenden Wort $x_i \in X$ abhängt. Ist dies der Fall, kann $p(X, Y)$ durch eine Formel der Form von Formel (1) beschrieben werden.

$$p(X, Y) = \prod_{t=1}^{\#X} p(y_t | y_{t-1}) * p(y_t | x_t)$$

Wahrscheinlichkeiten der Form von (1) können immer in die Form von Formel (2) überführt werden.

$$\begin{aligned} &\text{mit } \Theta_{i,j} \in \mathbb{R}^+, L = \text{alle moeglichen Labels}, W = \text{alle moeglichen Woerter}, \\ &X^* = \text{alle moeglichen Woerter-Vektoren}, Y^* = \text{alle moeglichen Label-Vektoren}, \\ &\text{und } Z = \sum_{X \in X^*} \sum_{Y \in Y^*} \prod_{t=1}^{\#X} \exp\left(\sum_{i,j \in L} \Theta_{i,j} * 1_{y_t=i} * 1_{y_{t-1}=j} + \sum_{i \in L, o \in W} \mu_{i,o} * 1_{y_t=i} * 1_{x_t=o}\right), \\ &p(X, Y) = \frac{1}{Z} \prod_{t=1}^{\#X} \exp\left(\sum_{i,j \in L} \Theta_{i,j} * 1_{y_t=i} * 1_{y_{t-1}=j} + \sum_{i \in L, o \in W} \mu_{i,o} * 1_{y_t=i} * 1_{x_t=o}\right) \end{aligned}$$

$1_{\text{Bedingung}}$ ist dabei eine Funktion, die genau dann Eins ist, wenn die Bedingung erfüllt ist und ansonsten Null. Je öfter $y_t = i$ und $y_{t-1} = j$ gemeinsam in der Verteilung vorkommen, desto größer wird $\Theta_{i,j}$ sein. Kommen $y_t = i$ und $y_{t-1} = j$ gar nicht gemeinsam vor, gilt

$\Theta_{i,j} * 1_{y_t=i} * 1_{y_{t-1}=j} = 0$. Gleiches gilt für $y_t = i$, $x_t = o$ und $\mu_{i,o}$. Die Thetas werden auch als *Gewichte* bezeichnet. Z bildet die Summe der Werte für alle möglichen Vorkommen von X und Y . Nach der Division durch Z ist somit sichergestellt, dass Formel (2) zwischen 0 und 1 liegt, sowie dass die Summe über alle möglichen Wahrscheinlichkeiten 1 ist, wie es für eine Wahrscheinlichkeitsverteilung der Fall sein muss.

Durch eine Abbildung der Indizes i, j und o auf k kann dies wiederum zu Formel (3) vereinfacht werden.

$$p(X, Y) = \frac{1}{Z} \prod_{t=1}^{\#X} \exp\left(\sum_{k=1}^K \Theta_k * f_k(y_t, y_{t-1}, x_t)\right),$$

$$\text{mit } Z = \sum_{X \in X^*} \sum_{Y \in Y^*} \prod_{t=1}^{\#X} \exp\left(\sum_{k=1}^K \Theta_k * f_k(y_t, y_{t-1}, x_t)\right)$$

Wie in Formel (4) zu sehen ist, kann jede multivariate Wahrscheinlichkeit in eine bedingte Wahrscheinlichkeit umgerechnet werden.

$$p(Y|X) = \frac{p(X, Y)}{\sum_{Y' \in Y^*} p(Y', X)}$$

Eine nahe liegende Art zu Orakeln ist nun Formel (5). Nebenbei sei erwähnt, dass ein so antrainiertes Orkal als ein Hidden Markov Model (HMM) bezeichnet wird.

$$\text{orakel}(X) = \text{argmax}_Y(p(Y|X))$$

Aufgrund des vereinfachten Modells, dass jedes Label y_i nur vom vorherigen Label y_{i-1} und dem entsprechenden Wort x_i abhängt, ist Formel (1) nur eine Annäherung an $p(X, Y)$ und somit sind auch alle weiteren Umformungen nur eine Annäherung. Ein CRF vereinfacht das Modell nun weiter, in dem es einige Θ_k und entsprechende f_k im Model weglässt. Durch das Weglassen von vielen kleinen Θ_k kann ein CRF die Berechnungszzeit beschleunigen, wobei die berechnete Verteilung und somit das spätere Orakeln nur marginal verändert werden. Auch erlaubt ein CRF negative Werte für Θ_k . Der Intuition nach sagen diese aus, dass ein gemeinsames Vorkommen von X und Y sehr unwahrscheinlich ist und geben somit Strafpunkte für die argmax_Y -Orakel-Funktion. Des Weiteren können in einem CRF sogenannte *custom feature functions* $f_{k'}(y_t, y_{t-1}, X)$ angegeben werden, die nicht nur vom entsprechenden Wort x_i sondern vom ganzen Wort-Vektor X abhängig sein können. Die $\Theta_{k'}$ zu diesen custom feature functions werden so berechnet, dass beim Orakeln der Trainings-Vektoren X die entsprechenden Label-Vektoren Y möglichst genau geraten werden, unter der Annahme, dass das Ergebnis berechnet werden muss und nicht einfach nachgeschaut werden darf. Die Praxis zeigt, dass ein CRF mit guten custom feature functions besser orakelt als ein HMM. Typische custom feature functions wären z. B.:

- $f_{k'_1}(y_t, y_{t-1}, X) = 1_{x_t \text{ ist ein Nomen}}$
- $f_{k'_2}(y_t, y_{t-1}, X) = 1_{x_t \text{ ist in einem Domänen-spezifischen Woerterbuch}}$
- $f_{k'_3}(y_t, y_{t-1}, X) = 1_{x_t \text{ ist in einem Domänen-spezifischen Woerterbuch und } x_{t-1} \text{ ist ein Nomen}}$

Abschließend seien zu CRF noch eine überraschende und eine gute Bemerkung erwähnt. Die Überraschende ist folgende: Aufgrund der Vereinfachung des Modells wie auch den custom feature functions ist es möglich, dass ein CRF ein Wort-Vektor, der in den Trainingsdaten enthalten war, falsch orakelt. Die gute Bemerkung ist, dass sich die erste Befürchtung eines

Informatikers nicht bestätigt, dass die Funktion $oracle(X) = \operatorname{argmax}_Y(p(Y|X))$ in Laufzeit von $O(\#L^{\#X})$ berechnet werden muss. Stattdessen kann sie durch dynamisches Programmieren in $O(\#L^2 * \#X)$ berechnet werden.

Implementierung der NYT

Die Rezepte der Kochseite der NYT haben alle eine Zutatenliste. Jede Zeile der Zutatenliste ist in einer Datenbank abgespeichert, aufgespalten nach der Zutat, einer Mengenangabe, der Mengeneinheit und Anderem. In dieser Datenbank sind mehr als 130.000 solcher Einträge, welche alle als Trainingsdaten für das CRF genutzt werden können. Die Zutat erhält dabei das Label *NAME*, die Mengenangabe *QTY* und die Einheit *UNIT*. Nach dem IOB2-Format erhält das erste Wort eines Labels den Prefix B-. Folgende Wörter erhalten den Prefix I-. Als CRF-Bibliothek verwendet die NYT CRF++. Abb. 2 bis 4 zeigen alle von Greene in CRF++-Format angegebenen custom feature functions.

Abb. 2: Trainingsdaten mit Feature Labels

3/4	I1	L12	NoCAP	NoPAREN	B-QTY
pound	I2	L12	NoCAP	NoPAREN	OTHER
shiitake	I3	L12	NoCAP	NoPAREN	B-NAME
mushrooms	I4	L12	NoCAP	NoPAREN	I-NAME
,	I5	L12	NoCAP	NoPAREN	OTHER
stemmed	I6	L12	NoCAP	NoPAREN	B-COMMENT
and	I7	L12	NoCAP	NoPAREN	I-COMMENT
quartered	I8	L12	NoCAP	NoPAREN	I-COMMENT

Die Trainingsdaten in Abb. 2 enthalten neben dem Wort-Vektor (1. Spalte) und dem Label-Vektor (letzte Spalte) eine Reihe von Feature-Labels (2.-5. Spalte). Die 2. Spalte gibt zu jedem Wort den Index des Wortes im Satz an. Die 3. Spalte klassifiziert die Sätze nach der Anzahl ihrer Wörter in Vierer-Abständen. Sätze der Länge 0-3 werden als L0 klassifiziert, Sätze der Länge 4-7 als L4, usw. Die 3. Spalte gibt an, ob das jeweilige Wort mit einem großen Buchstaben anfängt und die 4. Spalte ob, das Wort im Satz innerhalb von Klammern steht.

Die vom CRF zu verwendenden Features werden durch das Template in Abb. 3 angegeben. Ein Template hat das Format `UID:%x[Zeile,Spalte]`. Das B-Template entspricht der Wahrscheinlichkeit $p(y_t|y_{t-1})$, dass auf das Label y_{t-1} das Label y_t folgt. Abb. 4 zeigt die aus den Templates abgeleiteten Features, wenn das aktuelle Wort in Abb. 2 „mushrooms“ ist.

Aus den Trainingsdaten mit den Feature-Labels und den Templates generiert CRF++ custom feature functions wie in Abb. 5 exemplarisch gezeigt ist.

Abb. 3:

Feature Templates

```
# Unigram
U00:%x[-2,0]
U01:%x[-1,0]
U02:%x[0,0]
U03:%x[1,0]
U04:%x[2,0]
U05:%x[0,1]
U06:%x[0,2]
U07:%x[0,3]

U08:%x[-2,4]
U09:%x[-1,4]
U10:%x[0,4]
U11:%x[1,4]
U12:%x[2,4]

U13:%x[0,0]/%x[0,2]
U14:%x[0,1]/%x[0,2]
U15:%x[0,0]/%x[0,3]
U16:%x[0,0]/%x[0,4]
U17:%x[0,0]/%x[0,1]

# Bigram
B
```

Abb. 5: Beispiele für von CRF++ aus Abb. 2 und 3. extrahierte custom feature functions

U02:%[0,0]	$\rightarrow 1_{y_t = I - \text{Name und } x_t = \text{mushrooms}}$
U02:%[-2,4]	$\rightarrow 1_{y_t = I - \text{Name und } x_{t-2} \text{ 4. Spalte ist NoPAREN}}$
U13:%x[0,0]/%x[0,2]	$\rightarrow 1_{y_t = I - \text{Name und } x_t = \text{mushrooms und } x_{t-2} \text{ 2. Spalte ist NoPAREN}}$
B	$\rightarrow 1_{y_t = i \text{ und } y_{t-1} = j} \forall i, j \in L$

Abb. 4: Anwendung der Templates

pound
shiitake
mushrooms
,
stemmed
I4
L12
NoCAP
NoPAREN
NoPAREN
NoPAREN
NoPAREN
NoPAREN
mushrooms/L12
I4/L12
mushrooms/NoCAP
mushrooms/NoPAREN
mushrooms/I4
I - NAME B - Name

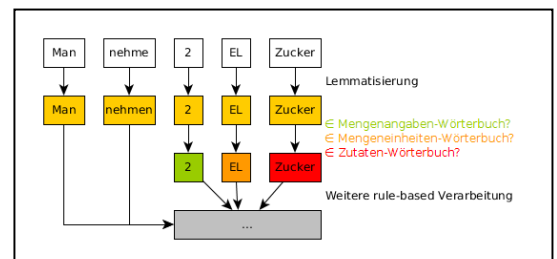
Diese CRF Implementierung berücksichtigt für ihre custom feature functions also nur die Position im Satz, die Länge des Satzes, ob ein Wort groß geschrieben wird und ob es eingeklammert ist. Bei einem Test mit 481 Rezepten orakelt dieses CRF 89% aller Zeilen einer Zutatenliste korrekt. (Greene, 2015) testet nur 481 Rezepte, obwohl er über 130.000 zur Verfügung hat, da er keine Möglichkeit zur automatischen Auswertung gefunden hat. Wenn das CRF was anderes orakelt hat, als in der Datenbank stand, dann war manchmal das Orakelte falsch, manchmal aber auch die manuell aufgebaute Datenbank und manchmal waren sogar beide falsch.

Dictionary- and rule-based

Die beiden unabhängigen Arbeiten von (Zeeshan, 2009) und (Wiegand et al., 2012) kategorisieren wir als *dictionary- and rule-based*. Die Grundidee ist in Abb. 6 zu sehen.

Sämtliche gesuchten *Entities* sind in einem Wörterbuch gespeichert. Da wir uns auf die Koch-Domäne beschränken, ist die Annahme eines solchen Wörterbuches vertretbar. Für uns wäre dass z. B. ein Wörterbuch, welches alle Zutaten enthält, ein Weiteres für alle Mengenangaben und ein Drittes für alle Mengeneinheiten. Um diese Wörterbücher möglichst klein zu halten, werden nur die Lemmata der gesuchten Entities gespeichert. Um nun Informationen aus einem Satz zu extrahieren, wird der Satz zuerst lemmatisiert und die einzelnen Lemmata

Abb. 6: Grundidee von dictionary- and rule-based extraction



danach in den Wörterbüchern nachgeschaut. Ohne die Lemmatisierung könnte beispielsweise bei „*Man zerkocht die Äpfel*“ Äpfel nicht extrahiert werden, obwohl *Apfel* im Wörterbuch steht. Die auf diese Weise extrahierten Entities können anschließend noch weiterverarbeitet werden, was wir als *rule-based* Weiterverarbeitung bezeichnen.

(Zeeshan, 2009) möchte aus Rezepten die Zutaten extrahieren und herausfinden, wie diese verarbeitet werden. Zum Beispiel ob sie gekocht, oder gebacken werden. Nachdem er mit einem Wörterbuch von Zutaten und einem von Verarbeitungs-Verben die gesuchten Entities gefunden hat, muss er noch verbinden, welche Verarbeitungs-Verben zu welchen Zutaten gehören. Er hat dafür zwei unterschiedliche Ansätze:

Der erste Ansatz verwendet eine **kontextfreien Grammatik**. Abb. 7 verdeutlicht das Prinzip beispielhaft. Die Terminal-Symbole der Grammatik sind Part-of-Speech-Tags sowie Tags für die bereits gefundenen Entities. Die Ableitungsregeln der Grammatik haben alle eine vordefinierte Bedeutung, wie z. B. *wende das Verarbeitungs-Verb auf alle Zutaten in diesen Satz an*. Ein Satz wird nun wie folgt bearbeitet: Zunächst wird er in Terminal-Symbole übersetzt. Anschließend wird geschaut, durch welche Regeln in der Grammatik diese Terminal-Symbole abgeleitet werden können. Wurde eine gültige Ableitung gefunden, wird abschließend die vordefinierte Bedeutung der Ableitung auf den Satz angewendet. Das Beispiel in Abb. 7 kommt so zu dem Ergebnis, dass bei dem Satz *Koche Erbsen und Bohnen* die *Erbsen* und die *Bohnen gekocht* werden sollen.

Der zweite Ansatz basiert auf **dependency-based parsing**. Abb. 8 zeigt ein exemplarisches Ergebnis eines dependency-based parsings des Satzes *Koche die Erbsen und Bohnen*. Aus diesem wird sofort ersichtlich, dass sich das Verb *kochen* auf die Objekte *Erbsen* und *Bohnen* bezieht. Das dependency-based parsing kann von Bibliotheken wie dem Stanford Parser erledigt werden.

Abb. 7: Beispielhafte kontextfreie Grammatik-basierte Weiterverarbeitung

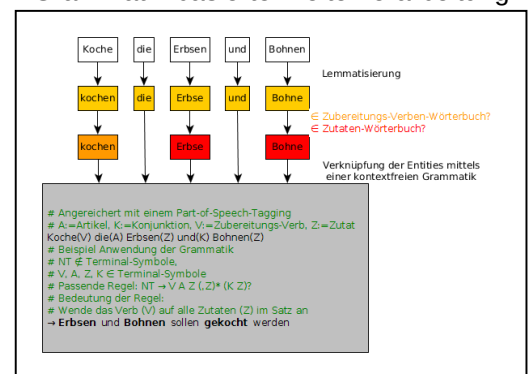
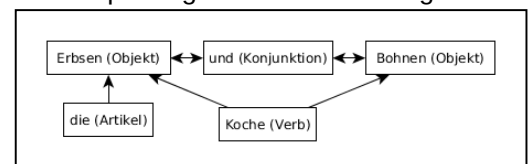


Abb. 8: Beispielhafte dependency-based parsing Weiterverarbeitung



Beide Ansätze werden anhand von 43 Rezepten evaluiert. Tabelle 1 zeigt die Ergebnisse. Die Precision zeigt, dass nahezu alle extrahierten Entities und ihre Beziehungen zu-

Tabelle 1

einander korrekt sind. Allerdings werden nur gut die Hälfte aller relevanten Informationen extrahiert, wie der Recall zeigt. Die Schlussfolgerung von (Zeeshan, 2009) ist, dass das dictionary-based Extrahieren nahezu perfekt funktioniert. Wenn bei der rule-based Weiterverarbeitung eine Regel der kontextfreien Grammatik erfolgreich angewendet werden kann, sind die extrahierten Informationen ebenfalls korrekt. Allerdings sind viele Informationen nicht von einer Regel abgedeckt und werden somit nicht extrahiert. Gleiches gilt für das dependency-based parsing. Es werden zwar ein wenig mehr korrekte Informationen erfasst, dafür extrahiert dieser Ansatz auch einige zusätzliche Informationen, die falsch sind.

	Precision	Recall
Mit kontextfreier Grammatik	97,39%	51,54%
Mit dependency-based parsing	95,40%	64,12%

(Wiegand et al., 2012) sind an folgenden vier Fragen interessiert: *Welche Lebensmittel passen zu welchem Anlass?*, *Welche Lebensmittel werden oft kombiniert?*, *Welche Lebensmittel können untereinander ausgetauscht werden?* und *Welche Lebensmittel sind Teil eines Gerichts*. Um diese Fragen zu beantworten, extrahieren sie Informationen aus Wikipedia und dem Forum von Chefkoch.de. Sie arbeiten also mit rein textuellen Daten und können beispielsweise nicht auf der Struktur eines Rezeptes mit Zutatenliste aufbauen. Für uns relevant sind ihre Methoden *Surface Patterns* und *Statistical Co-occurrence*.

Surface Patterns ist von der Idee her identisch mit der zuvor vorgestellten kontextfreien Grammatik von (Zeeshan, 2009). (Wiegand et al., 2012) stellen ebenfalls fest, dass die extrahierten Informationen meist korrekt sind, aber viele Informationen nicht gefunden werden. Im Gegensatz zu (Zeeshan, 2009) schreiben sie jedoch, dass sie mit einer dependency-based parsing Weiterverarbeitung nicht mehr korrekte Informationen extrahieren konnten. Sie führen das auf systematische Fehler bestehender Parser für die deutsche Sprache zurück. Diese sind mit Zeitungsartikel antrainiert, in welchen im Allgemeinen anders formuliert wird als in der Koch-Domäne.

Statistical Co-occurrence sucht nach gemeinsamen Vorkommen von Entities. Wenn beispielsweise ein Lebensmittel (Popcorn) mit einem Anlass (Kinobesuch) im gleichen Text steht, wird extrahiert, dass das Lebensmittel zu diesem Anlass passt (Popcorn passt zum Kinobesuch).

Das Ergebnis der Extraktion ist je eine Liste von Lebensmittel für jede der vier Fragen. Da eine Liste von Ergebnissen nicht passend mit Precision und Recall ausgewertet werden kann und die Evaluierung keine weiteren Erkenntnisse für diese Arbeit bringt, sei für die Evaluierung auf die originale Ausarbeitung verwiesen.

Automatisches Auszeichnen mit *cueML*

CRF-
based
Prototyp

Dictionary-
and rule-
based
Prototyp



Inspiziert von dem vorherigen Abschnitt Information Extraction in der Koch-Domäne stellen wir hier zwei Prototypen vor, um aus den Zubereitungs-Anweisungen von unserem Kochbuch eine Zutatenliste zu extrahieren. Da wir auf keine vorhandene Semi-Struktur aufbauen können, ist ein Regular Expression-based Ansatz nicht möglich. Wir implementieren einen Conditional Random Field-based (CRF-based) und einen dictionary- and rule-based Prototypen. Der CRF-based Prototyp ist sehr simpel. Uns ist bereits während der Konstruktion klar geworden, dass wir andere Voraussetzungen als (Greene, 2015) haben und dies daher nicht der richtige Ansatz für uns ist.

CRF-based Prototyp

Unser CRF-based Prototyp ist in unserem Git-Repository unter dem Tag CRF-BasedPrototypeV0.1 zu finden. Er verwendet das SWIG Interface für *Python 3* von CRFSuite 0.12.

Wie in CRF erklärt, muss der Prototyp zuerst antrainiert werden. Abb. 1 zeigt einen kleinen Ausschnitt unserer Trainingsdaten für den CRF-based Prototypen. Nebenbei sei erwähnt, dass wir diese Anweisungen heute als eine Mehlschwitze bezeichnen würden.

Aus diesem kleinen Ausschnitt der Trainingsdaten sind bereits viele Problematiken ersichtlich, was später in der Evaluierung dieses Prototypen genauer erläutert wird. Beispielsweise wird aus den Labels nicht ersichtlich, dass die Mengenanabe „eine“ zu der Zutat „Zwiebel“ gehört. Daher haben wir zunächst einen simplen Prototypen gebaut, um zu schauen, ob CRF wirklich der richtige algorithmische Ansatz für unser Problem ist. Wie in Abb. 1 haben wir 10 Rezepte gelabelt. Mit 9 davon haben wir den CRF-based Prototypen trainiert und an dem 10. getestet. Als features verwenden wir ausschließlich die Wort-Identität, sowie den Übergang von zwei benachbarten Labels (B-Template). Für die Gewichte der features erlauben wir auch negative Werte.

Abb. 1: Ausschnitt
unserer CRF
Trainingsdaten

Man	0
Schmort	0
eine	B-Quantity
fein	0
geschnittene	0
Zwiebel	B-Ingredient
mit	0
reichlich	0
frischer	0
Butter	B-Ingredient
,	0
läßt	0
ein	B-Quantity
bis	I-Quantity
zwei	I-Quantity
Eßlöffel	B-Unit
voll	0
feines	0
Mehl	B-Ingredient
darin	0
anziehen	0
,	0
und	0
[...]	0

Dictionary- and rule-based Prototyp

Nachdem wir zu der Erkenntnis gekommen sind, dass der vorherige Ansatz für diese Arbeit nicht zielführend ist, stellen wir hier einen dictionary- and rule-based Prototypen vor. Unsere Lemmatisierung für den dictionary-based Teil baut auf *treetagger* Version 1.0.1 auf.

Version 0.1

Ein erster Prototyp ist in unserem Git-Repository unter dem Tag Dict-AndRule-BasedPrototypeV0.1 zu finden. Das Ziel dieses Prototypen ist, erst einmal ausschließlich zu testen, ob das dictionary-based Extrahieren unserer gesuchten Entities (Zutaten, Mengenangaben und Einheiten) funktioniert. Deswegen ist er nach dem Motto *quick and dirty* implementiert. Nachdem die erste Evaluierung positiv stimmt, haben wir den Prototypen quasi weggeworfen und in der folgenden Version 0.2 redesigned, wie es sich für einen quick and dirty Prototypen gehört. Daher wird hier auch nicht weiter auf Implementierung-Details eingegangen sondern nur auf konzeptionelle Ideen.

Abb. 2 zeigt den nahezu identischen Ablauf der dictionary-based Extraktion aus der vorherigen Sektion. Die Überschrift eines Rezeptes wird als normaler Satz gesehen. Sie muss bei der Extraktion berücksichtigt werden, da manchmal eine Zutat nur in dieser erwähnt wird. Die Sätze splitten wir anhand von Satzzeichen. Um beispielsweise aus *Nach Nro. 2 wird [...]* nicht zwei Sätze zu machen, haben wir eine Menge von Abkürzungen definiert, bei denen der Split wieder zusammengeführt wird.

In Abb. 3 ist in einem ersten Schritt der Output eines Satzes von TreeTagger zu sehen. Neben dem Lemma gibt er zusätzlich ein Part-of-Speech-Tag (PoS) zu jedem Wort aus. Wie bei *Sellerie-* und *Scorzonerwurzeln* kann TreeTagger allerdings manchmal kein Lemma finden. Deswegen nehmen wir zuerst von jedem Wort ohne Lemma die Wort-Identität als Lemma. Anschließend versuchen wir bei jedem abgekürzten Wort die Endung zu finden. Abgekürzte Wörter haben das PoS TRUNC für truncation. Da alle abgekürzten Wörter, an denen wir interessiert sind, Zutaten und somit Nomen sind, suchen wir das nächste Nomen im Satz, welches eine typisch abgekürzte Endung hat. Ein Beispiel für so eine Endung ist *wurzel*. Ist dies der Fall, wird in dem abgekürzten Lemma der Bindestrich durch die entsprechende Endung ersetzt.

Was jetzt noch fehlt, sind die einzelnen Wörterbücher der zu extrahierenden Entities:

- In unserer cueML-Schema Definition ist eine Liste von erlaubten **Mengeneinheiten** (unit). Diese Liste von erlaubten Werten nehmen wir als Wörterbuch der Einheiten.
- Für die **Mengenangaben** verwenden wir kein echtes Wörterbuch sondern eine Funktion. Diese bekommt als Argument ein Lemma und returniert in folgenden drei Fällen *True*:

Abb. 2: Dictionary-based Extraktion der Entities

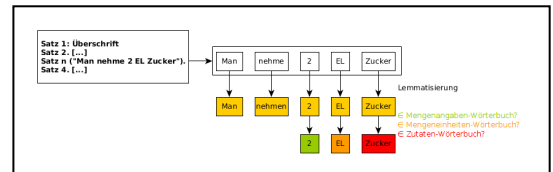
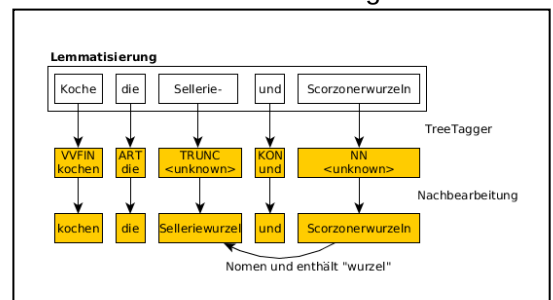


Abb. 3: Nachbearbeitung von TreeTaggers Lemmatisierung



- Das Lemma ist eine Zahl.
- Das Lemma hat die Form *Zahl-Zahl*.
- Das Lemma ist in einer Menge von vordefinierter Mengen-Wörter (*ein, eine, einige, ...*).

Ansonsten returniert die Funktion *False*.

- Wie in Domänen-spezifisches Vokabular cueML erläutert, haben wir alle ausgezeichneten Zutaten mittels einer Liste von *ingredient*-Elementen an den BLS angeschlossen. Unten stehende Abb. 4 zeigt, wie wir aus diesen ein **Wörterbuch von Zutaten** extrahieren. Jedes Nomen einer Basisform ist ein Eintrag im Wörterbuch. Zu jedem Eintrag speichern wir zusätzlich eine Liste von möglichen *xml:ids* ab. Wein könnte z. B. Weißwein wie auch Rotwein sein. Bei einer rule-based Weiterverarbeitung kann aus der Liste der möglichen *xml:ids* ggf. eine eindeutige *xml:id* bestimmt werden. Dies werden wir in der Version 0.2 in Angriff nehmen. Der letzte Eintrag in Abb. 4b ist besonders. Frau Davidis verwendet oft Fleisch als Zutat und aus dem Kontext wird ersichtlich, was für ein Fleisch gemeint ist. Bei einer Rindfleischsuppe zum Beispiel ist mit Fleisch ziemlich sicher Rindfleisch gemeint. Um auch das Lemma Fleisch als Zutat extrahieren zu können, fügen wir dieses dem Wörterbuch hinzu. Der hinterlegte Wert ist eine Liste von allen *xml:ids*, in dessen Einträgen das Wort Fleisch vorkommt.

Die Einschränkung auf das Nomen einer Zutat ist eine große Beschleunigung der Laufzeit der dictionary-based Extraktion. Bei einem Satz der Länge n werden so n *look ups* im Wörterbuch benötigt. Um Zutaten der Länge eins und zwei nachschlagen zu können, wären bereits $n * (n-1)$ *look ups* nötig. Problematisch wären auch Fälle, in denen die Wörter einer Zutat nicht benachbart sind wie z. B. *Man nehme Wein; bevorzugt Weißen*.

Wenn ein Lemma nicht im Zutaten-Wörterbuch ist, überprüfen wir zusätzlich, ob es eine zusammengesetzte Zutat ist, bzw. ein eigenes Rezept. Ist im Wort eins der Wörter *bouillon, klöße oder kloß* enthalten, ist dies auch ein positives Ergebnis und es wird eine leere Liste von möglichen BLS zurückgegeben.

Abb. 4: Extraktion eines Zutaten-Wörterbuches aus den cueML *ingredient*-Elementen

Abb. 4a: Beispielhafter Ausschnitt von cueML *ingredient*-Elementen

```
<cue:listIngredient xmlns="http://www.tei-c.org/ns/1.0" xmlns:cue="cueML">
  <cue:ingredient xml:id="Rindkochfleisch" BLSref="U180100">
    <cue:prefBasicForm>Rindfleisch</cue:prefBasicForm>
  </cue:ingredient>
  <cue:ingredient xml:id="Middel" BLSref="V582100">
    <cue:prefBasicForm>Middel</cue:prefBasicForm>
    <cue:altBasicForm>Kalbsmieder</cue:altBasicForm>
    <cue:note>Kalbsmieder ist auch unter dem Synonym [...]</cue:note>
  </cue:ingredient>
  <cue:ingredient xml:id="Weißwein" BLSref="P220000">
    <cue:prefBasicForm>Weißwein</cue:prefBasicForm>
    <cue:altBasicForm>weißer Wein</cue:altBasicForm>
  </cue:ingredient>
  <cue:ingredient xml:id="Butter" BLSref="Q610000">
    <cue:prefBasicForm>Butter</cue:prefBasicForm>
  </cue:ingredient>
  <cue:ingredient xml:id="Rotwein" BLSref="P240000">
    <cue:prefBasicForm>Rotwein</cue:prefBasicForm>
    <cue:altBasicForm>roter Wein</cue:altBasicForm>
  </cue:ingredient>
  <cue:ingredient xml:id="Kalbkochfleisch" BLSref="U380100">
    <cue:prefBasicForm>Kalbfleisch</cue:prefBasicForm>
  </cue:ingredient>
  [...]
</cue:listIngredient>
```

Abb. 4b: Aus Abb. 4a extrahiertes Zutaten-Wörterbuch

```
{
  "Rindfleisch" : "Rindkochfleisch",
  "Middel" : "Middel",
  "Kalbsmieder" : "Middel",
  "Weißwein" : "Weißwein",
  "Wein" : [
    "Weißwein",
    "Rotwein"
  ],
  "Butter" : "Butter",
  "Rotwein" : "Rotwein",
  "Kalbfleisch" : "Kalbkochfleisch",
  "Fleisch" : [
    "Rindkochfleisch",
    "Kalbkochfleisch"
  ],
  [...]
}
```

Version 0.2

Die Evaluierung der Version 0.1 zeigt, dass die dictionary-based Extraktion funktioniert. Daher lohnt sich die Weiterverfolgung des dictionary- and rule-based Ansatzes. Bevor wir uns jedoch mit der rule-based Weiterverarbeitung beschäftigen, erinnern wir uns daran, dass der Prototyp bis jetzt eine quick and dirty Machbarkeitsanalyse war, welche wir zuerst redesignen wollen.

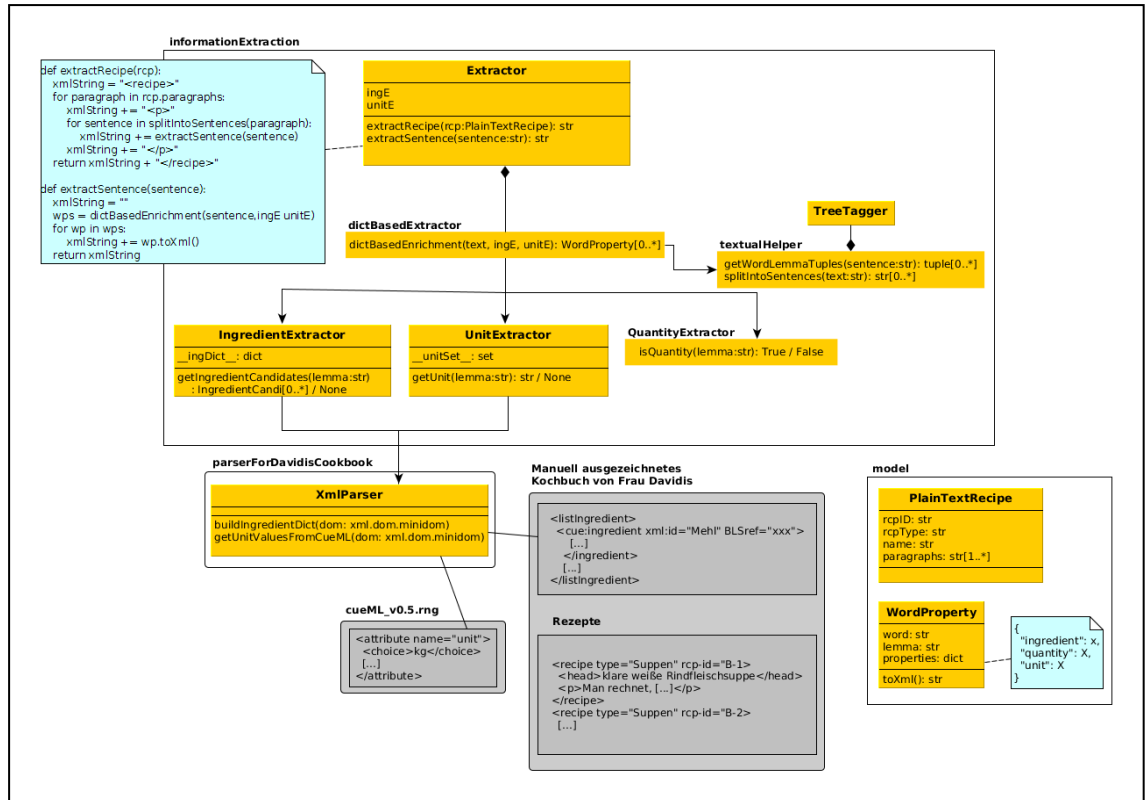
Design der dictionary-based Extraktion

Abb. 5 zeigt die aufs wesentliche reduzierten Bestandteile des Designs. Jedes Rezept wird zunächst in ein **PlainTextRecipe** umgewandelt, welches zu jedem Rezept die rcp-id, den Rezept-Typ, den Namen und eine Liste von Zubereitungs-Anweisungen speichert. Satzweise führt der **Extractor** dann mittels **dictBasedEnrichment** eine Extraktion der gesuchten Entities durch. Dazu werden zunächst mit der Hilfe von **TreeTagger** die Lemmata zu jedem Wort gebildet. Diese Lemmata werden dann durch den **IngredientExtractor**, **UnitExtractor** und **QuantityExtractor** getestet. Ist einer dieser Tests positiv, wird in der entsprechenden **WordProperty** im **properties**-dict das Ergebnis des Tests hinterlegt. Der **QuantityExtractor** besteht nur aus der Funktion **isQuantity**, welche ein Lemma bekommt und *True* bzw. *False* wiedergibt. Die Wörterbücher für den **IngredientExtractor** sowie den **UnitExtractor** werden mittels dem **XmlParser**, wie zuvor beschrieben, aus dem manuell mit cueML angereicherten Kochbuch extrahiert.

Das Ergebnis der Extraktion des Satzes „Schütte 1 Maß Wein hinzu.“ mit dem Zutaten-Wörterbuch aus Abb. 4b wäre zum Beispiel die Liste folgender WordProperties:

```
[
  WordProperty(word="Schütte", lemma="schütten", properties={} ),
  WordProperty(word="1", lemma="1", properties={"quantity":"1"} ),
  WordProperty(word="Maß", lemma="Maß", properties={"unit":"Maß"} ),
  WordProperty(word="Wein", lemma="Wein", properties={"ingredient":["Weißwein,
    "Rotwein"]}) ,
  WordProperty(word="hinzu", lemma="hinzu", properties={} ),
  WordProperty(word=".", lemma=".", properties={} )
]
```

Abb. 5: Design der dictionary-based Extraktion



Das saubere Design erleichtert auch eine genauere Analyse der vorherigen Evaluation der Version 0.1. Aufbauend auf dieser Analyse haben wir noch folgende Punkte verbessert:

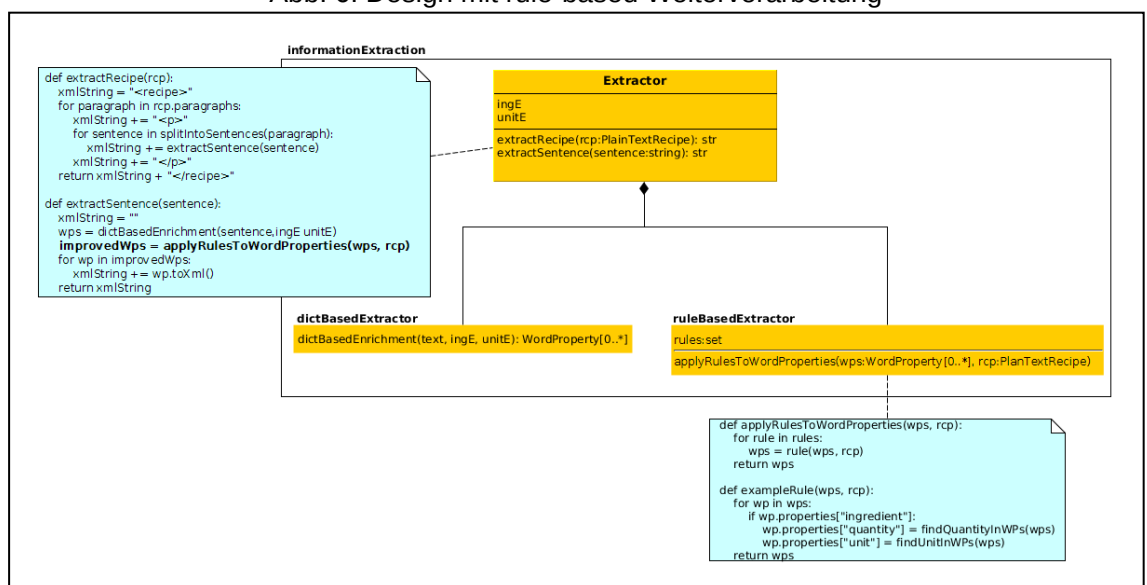
- Um zu vermeiden, dass beispielsweise *abgebrüht* als Zutat extrahiert wird, da es das Teilwort *brühe* enthält, schlägt der Test mit dem IngredientExtractor nun bei klein geschriebenen Wörtern immer fehl.
- TreeTagger kann zu Pluralen die mit *n* gebildet werden, oft das Lemma nicht bilden (z. B. bei *Scorzonnerwurzeln* oder *Kartoffeln*). Daher überprüfen wir beim Test mit dem Zutaten-Wörterbuch in der Methode *dictBasedEnrichment* das Lemma sowie das Lemma ohne *n* am Ende, wenn es auf *n* aufhört.
- TreeTagger findet manchmal mehrere Lemmata durch | getrennt. Beispielsweise kann das Lemma zu „Linsen“, „Linse|Linsen“ sein, was nicht in unserem Zutaten-Wörterbuch zu finden ist. Ist dies der Fall, nehmen wir daher nur das erste Lemma (Linse).
- Manchmal schreibt Frau Davidis eine Zutat als zusammengesetztes Wort (z. B. *Petersilien-Wurzel*) und manchmal nicht (*Petersilienwurzel*). Man könnte die zusammengesetzte Schreibweise ins Wörterbuch als alternative Schreibweise hinzufügen. Wir berücksichtigen das hingegen im Programm. Wenn das Lemma zu einem Wort nicht gefunden werden konnte, eliminieren wir den Bindestrich aus dem Wort, und versuchen dann erneut ein Lemma zu finden.
- Die Liste der typischen Endungen von abgekürzten Wörtern wurde um *klöße*, *kloß*, *bröt* und *brod* erweitert.
- Uns sind einige Fehler im manuell ausgezeichneten Kochbuch aufgefallen, welches unseren Golden Standard bildet, die wir korrigiert haben.

Rule-based Weiterverarbeitung

In das saubere Design aus Abb. 5 fügen wir nun eine rule-based Weiterverarbeitung ein. Diese ist in Abb. 6 zu sehen. Nachdem die *WordProperties* extrahiert sind, wird auf diese eine Menge von Regeln angewendet. Eine Regel bekommt als Parameter die Liste der *WordProperties* eines Satzes, sowie das Rezept, zu dem der Satz gehört und gibt eine Liste mit geänderten *WordProperties* zurück. Beispiele für solche Regeln sind:

- **findQuantityAndUnitOfIngredientRule:** Die Regel versucht zu jeder *WordProperty* (*wp*), die als Zutat ausgezeichnet ist (*key "ingredient" ∈ wp.properties*), eine Mengenangabe und Einheit zu finden. Sie sucht dafür den nächsten Nachbarn von links (*wp'*), dessen *WordProperty* als Mengenangabe bzw. Einheit ausgezeichnet ist und fügt diese der *wp* hinzu (*wp.properties["quantity"] = wp'.properties["quantity"]*). Um zu verhindern, dass beispielsweise bei „Man nehme 2 EL Mehl und Butter“ 2 EL Butter extrahiert wird, bricht diese Regel ab, wenn vor der Mengenangabe bzw. Einheit eine andere Entity wie eine andere Zutat oder Kochzeit gefunden wird.
- **dissolveAmbiguityWein:** Wenn das Lemma zu einer Zutat *wein* enthält, sucht diese Regel nach den Wörtern „rot“ oder „weiß“. Wird eins von beiden gefunden, werden aus der Liste der möglichen *xml:ids* alle herausgefiltert, die dieses Wort nicht enthalten.
- **dissolveAmbiguityFleisch:** Ist die Zutat *Fleisch*, der Rezept-Typ *Suppen* und „rind“ ist im Namen des Rezeptes enthalten, so wird die *xml:id* auf Rindkochfleisch gesetzt.
- **dontUseRule:** Wenn vor einer Zutat das Wort „ohne“ steht, wird die Zutaten-Eigenschaft des Wortes zurückgenommen (*wp.properties["ingredient"] = None*).
- **optionalRule:** Sind nach einer Zutat die Lemmata „können“ und „wegbleiben“ im Satz enthalten, so wird diese Zutat als optional ausgezeichnet (*wp.properties["optional"] = True*).
- **altGrpRule:** Folgt auf eine Zutat das Lemma „statt“, werden alle Folgenden durch „und“ separierten Zutaten als Alternative zu der ersten Zutat ausgezeichnet.

Abb. 6: Design mit rule-based Weiterverarbeitung



Automatisches Auszeichnen mit cueML

Nachdem alle Entities mittels den *WordProperties* extrahiert wurden, müssen diese noch, mit dem *cueML*-Vokabular ausgezeichnet, zurückgeschrieben werden. Aufgrund unserer guten Kapselung der Entities in den *WordProperties* ist dies trivial. Jede *WordProperty* hat

eine *toXml*-Methode. Ist ein Wort keine Zutat, wird von der Methode die Wort-Identität zurückgegeben. Ansonsten wird die Word-Identität in ein *recipeIngredient*-Element mit den entsprechenden Attributen eingeschlossen.

Transformation in HTML

Durch eine Transformation der mit cueML ausgezeichneten Rezepten in HTML kann das digitalisierte Kochbuch nun übers Internet jedem zugänglich gemacht werden.

Das Python-Script *cueML2Json.py* parst die cueML Rezepte in ein *JSON-Objekt*, welches von einer Webseite ausgelesen werden kann. In der oberen Navigationsleiste im Menüpunkt Rezepte sind aufbauend auf dem von *cueML2Json.py* erstellten JSON-Objekt die in HTML transformierten Rezepte von Frau Davidis zu finden. Da unser Prototyp noch ausschließlich die Zutaten mit ihren Mengenangaben und Mengeneinheiten auszeichnet, kann die Transformation auch nur diese Entities berücksichtigen. Daher enthält die Webseite beispielsweise keine Verweise zwischen den Rezepten.

Eine andere Möglichkeit die mit cueML ausgezeichneten Rezepte in HTML zu transformieren ist mittels *XSLT*. Während meiner SHK-Tätigkeit habe ich die mit dem experimentellen cueML ausgezeichneten Rezepte mittels XSLT in HTML umgewandelt. Dies hat den Nachteil, dass die Erstellung der Navigationsbar für die Rezepte schwieriger ist, da nicht alle Rezepte in einem Objekt gekapselt sind. Da das Auszeichnen mit dem experimentelle cueML manuell erfolgte, wurden dort alle Entities wie beispielsweise Zubereitungszeiten oder Verweise auf andere Rezepte ausgezeichnet. Als Beispiel wie alle Entities von cueML in HTML dargestellt werden können, sei daher hier auf die Transformation des experimentellen cueMLs mittels XSLT verlinkt. Da sich das experimentelle cueML sehr von der aktuellen Version unterscheidet, ist eine Anpassung des Scripts jedoch leider aufwendig.

Evaluierung

Recall & Precision

CRF-based Prototyp

Dictionary- and rule-based Prototyp ▾

Auf dieser Seite werden die zuvor vorgestellten Prototypen evaluiert. Als Metriken verwenden wir die Recall und die Precision, welche zuerst erläutert werden.

Recall & Precision

Wir definieren Recall und Precision wie in den Formeln (1) und (2) nach (Hohto et al., 2005). Der Recall berechnet somit, ein wie großer Anteil der relevanten Informationen gefunden wurde. Die Precision berechnet, ein wie großer Anteil der extrahierten Informationen relevant ist. Je höher diese beiden Metriken sind, desto besser ist der evaluierte Algorithmus.

$$\text{Recall} = \frac{\#(\text{retrieved} \cap \text{relevant})}{\# \text{relevant}} \quad (1) \quad \text{Precision} = \frac{\#(\text{retrieved} \cap \text{relevant})}{\# \text{retrieved}}$$

Beide Metriken sind nur zusammen aussagekräftig. Ein perfekter Recall-Wert von eins könnte ansonsten erreicht werden, indem einfach alle Informationen als relevant extrahiert werden. Dies würde jedoch die Precision verschlechtern, sofern nicht alle Informationen relevant sind, wovon auszugehen ist. Ein sehr guter Wert für die Precision könnte hingegen dadurch erreicht werden, dass nur eindeutig als relevant klassifizierbare Informationen extrahiert werden. Dies verschlechtert wiederum den Recall, weil dadurch meist viele relevante Informationen nicht mehr gefunden werden. Würden gar keine Informationen extrahiert, wäre die Precision nicht definiert, da dann durch null geteilt wird.

CRF-based Prototyp

Antrainiert haben wir den CRF-based Prototypen mit diesen 9 Rezepten. Unser Test-Rezept besteht aus 96 Wörtern. 83 der entsprechenden 96 Labels sind *O* für *others*.

Tabelle 1 zeigt die Stellen, in denen sich die manuellen und die vom Prototypen extrahierten Labels unterscheiden. Die beiden Zutaten *Kartoffeln* und *Kartoffel-Klöße*, die nicht extrahiert wurden, sind auch nicht in den Trainings-Rezepten vorhanden. *Eine* und *einige* hingegen sind in den Trainingsdaten vorhanden. Sie sind mal als Mengenangabe gelabelt (z. B. in *eine geriebene gelbe Murzel*) und mal nicht (z. B. in *dies wird eine Weile gerührt*). Wenn dem CRF nur positive

Tabelle 1: Unterschiede zwischen den manuellen und extrahierten Labels

Wort im Satz	Manuelle Labels	Orakelte Labels
eine	B-Quantity	O
einige	B-Quantity	O
Kartoffeln	B-Ingredient	O
Kartoffeln	B-Ingredient	O
Kartoffel-Klöße	B-Ingredient	O

Tabelle 2:
Precision & Recall

	Precision	Recall
Mit negativen Gewichten	100,00%	61,54%
Ohne negativen Gewichten	100,00%	53,85%

Gewichte erlaubt sind, orakelt er auch *Griesmehl* als *O*, obwohl Griesmehl in den Trainingsdaten genau einmal vorkommt und dort das Label *B-Ingredient* hat.

Tabelle 2 zeigt die Precision und den Recall des CRF mit ausschließlich positiven, wie auch mit negativen Gewichten. Alle extrahierten Entities sind korrekt. Allerdings wurden 5 bzw. 6 der 13 Entities des Rezeptes nicht erkannt sondern als *O* gelabelt.

Das Antrainieren mit 9 Rezepten, sowie das Auswerten anhand nur einem Rezept ist natürlich kein Maßstab für eine echte Evaluierung. Allerdings sind uns bereits an dem simplen mit 9 Rezepten antrainierten Prototypen viele Stolpersteine aufgefallen.

Abb. 1: Zu Abb. 2 entsprechender mit cueML ausgezeichnete Text

Man schmort eine fein geschnittene `<cue:recipeIngredient ref="#Zwiebel" quantity="1">Zwiebel</cue:recipeIngredient>` mit reichlich frischer `<cue:recipeIngredient ref="#Butter" quantity="reichlich">Butter</cue:recipeIngredient>`, läßt ein bis zwei Eßlöffel voll feines `<cue:recipeIngredient ref="#Mehl" atLeast="1" atMost="2" unit="EL">Mehl</cue:recipeIngredient>` darin anziehen, und [...]

Abb. 1 zeigt den mit cueML ausgezeichneten Text zu den gelabelten Trainingsdaten aus Abb. 2. Bereits in diesem kleinen Ausschnitt der Trainingsdaten lassen sich viele Stolpersteine finden:

- „Ein bis zwei Eßlöffel“ sind in cueML eindeutig mit den Attributen `atLeast="1"` und `atMost="2"` abzubilden. **Ein eindeutiges Mapping zwischen cueML und den Labels aus Abb. 2 ist jedoch nicht trivial.** Aus (ein|B-Quantity) muss das Attribut `atLeast="1"` und aus (zwei|I-Quantity) das Attribut `atMost="2"` abgeleitet werden. Alternativ hätte man auch folgende Labels überlegen können; *ein* (*B-atLeast*) *bis* (*O*) *zwei* (*B-atMost*) oder *ein* (*B-atLeast*) *bis* (*B-IndicatorForAtLeastAtMost*) *zwei* (*B-atMost*).
- Ungeachtet davon, welche Labels beispielsweise für *ein bis zwei* verwendet werden, geht aus ihnen noch nicht hervor, dass sich diese Mengenangabe auf das *Mehl* bezieht. Wir stehen damit vor dem allgemeinen Problem, **dass mittels CRF zwar Entities extrahiert werden können, aber nicht Beziehungen zwischen diesen.** (Greene, 2015) hatte dieses Problem nicht. Wenn ein CRF-Orakel auf jede Zeile einer bestehenden Zutatenliste angewendet wird, ist klar, dass sich die extrahierte Mengenangabe auf die **eine** Zutat bezieht. Bei seinem Beispiel „4 tablespoons melted nonhydrogenated margarine, melted coconut oil or canola oil“ stellt er selber fest, dass sein CRF dafür nicht gerüstet ist („This ingredient phrase contains multiple ingredient names, which is a situation that is not accounted for in our database schema. We need to rethink the way we label ingredient parts to account for examples like this.“ (Greene, 2015)).
- Es ist schwierig aus dem cueML-Attribut `quantity="1"` abzuleiten, dass zu dem vorherigen Wort *eine* das Label *B-Quantity* gehört. Daher muss **das Labeln der Trainingsdaten manuell erledigt werden, obwohl es bereits mit cueML ausgezeichnete**

Abb. 2: Ausschnitt unserer CRF Trainingsdaten

Man	0
schmort	0
eine	B-Quantity
fein	0
geschnittene	0
Zwiebel	B-Ingredient
mit	0
reichlich	0
frischer	0
Butter	B-Ingredient
,	0
läßt	0
ein	B-Quantity
bis	I-Quantity
zwei	I-Quantity
Eßlöffel	B-Unit
voll	0
feines	0
Mehl	B-Ingredient
darin	0
anziehen	0
,	0
und	0
[...]	0

Rezepte gibt. Aus den 10 verwendeten Rezepten ergeben sich mehr als 1.500 Zeilen, die gelabelt werden müssen. Wie (Greene, 2015) den CRF mit mehr als 130.000 Trainingsdaten anzutrainieren, ist somit ein nicht zu stemmender Aufwand. Dies gilt insbesondere, da nach dem ersten Punkt die zu verwendeten Labels nicht eindeutig sind. Dementsprechend müssen unterschiedliche Labels für einen optimalen Algorithmus evaluiert werden und somit die Trainingsdaten sogar mehrmals zeitaufwendig mit unterschiedlichen Labels erstellt werden.

- Im Gegensatz zu (Greene, 2015) labeln wir ganze Sätze und nicht nur eine Zutatenliste. **Die meisten unserer Labels sind daher O für Other.** Dies führt dazu, dass das CRF-Orakel Wörter wie *eine* bevorzugt als *O* orakelt anstatt als *B-Quantity*.
- Daran das *Kartoffeln* nicht erkannt wird, wird auch deutlich, **dass der CRF nur Wörter als Zutaten erkennen kann, die auch in den Trainingsdaten waren.** Wäre in den Trainingsdaten *eine Kartoffel* vorhanden, so könnte der CRF *Kartoffeln* immer noch nicht erkennen. Daher ist als custom feature das Lemma des Wortes nötig. Wenn wir nun allerdings davon ausgehen, dass wir alle Zutaten (in den Trainingsdaten) kennen und zusätzlich die Lemmata der Wörter haben, können wir die Zutaten-Entities auch über einen Wörterbuch-Check extrahieren und brauchen den CRF dafür nicht.

Des Weiteren muss noch entwickelt werden, **wie die Unterscheidung zwischen Zutaten, optionalen Zutaten, alternativen Zutaten und nicht zu verwendeten Zutaten zu modellieren ist.** Für diese sind auf jeden Fall eigene Labels nötig. Da die Wörter/Zutaten zu den unterschiedlichen Labels jeweils die selben sind, braucht das CRF zur Unterscheidung wahrscheinlich zusätzlich weitere Labels und Features. Diese zu entwickeln und zu evaluieren ist jedoch wieder aufwendig.

Wir wollen damit nicht ausschließen, dass es möglich ist, die Zutaten mittels CRF zu extrahieren. Allerdings stößt dieser Algorithmus bei den schlecht strukturierten Zubereitungs-Anweisungen von Frau Davides auf viele Hindernisse und ist somit nicht der richtige Ansatz für uns. (Greene, 2015) konnte auf eine bereits im Rezept vorhandene Zutatenliste zurückgreifen sowie auf über 130.000 Trainingsdaten. Beides ist bei uns nicht gegeben. Stattdessen haben wir daher den *dictionary- and rule-based*-Ansatz weiterverfolgt.

Dictionary- and rule-based Prototyp

Unsere dictionary- and rule-based Prototypen testen wir anhand von Recall und Precision. Wir haben dazu die ersten 50 Suppen von Frau Davidis' Kochbuch manuell ausgezeichnet und verwenden dies als Golden Standard. Unsere Prototypen zeichnen die gleichen 50 Rezepte aus und werden dann gegen unseren Golden Standard evaluiert.

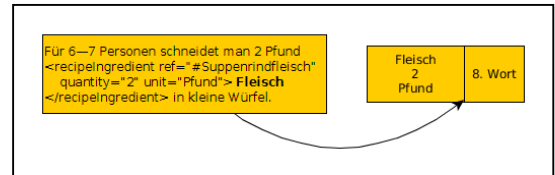
Version 0.1

Das Ziel dieses Prototypen ist es zu testen, ob unsere gesuchten Entities (Zutaten, Mengenangaben und Einheiten) extrahiert werden können. Da bei den Mengenangaben und Einheiten meine Intuition nicht Alarm schlägt, reichen mir ein paar Beispieldaten, um zu verifizieren, dass das funktioniert. Dies scheint der Fall zu sein. Daher wird eine genauere Evaluation der Mengenangaben und Einheiten auf einen ausgereifteren Prototypen verschoben. Folgend werden daher zunächst nur die Zutaten genauer betrachtet.

Abb. 3 zeigt, wie wir den **Recall** auswertet

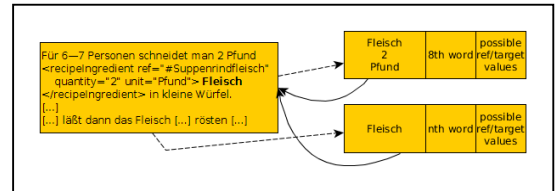
haben. Idealerweise ist in den extrahierten Zutaten jede Zutat wiederzufinden, die in unserem Golden Standard auch manuell ausgezeichnet wurde. Ist ein Wort im Golden Standard als Zutat ausgezeichnet, überprüfen wir einfach, ob unser Prototyp an der gleichen Stelle eine Zutat extrahiert hat.

Abb. 3: Recall



Die Auswertung der **Precision** ist komplexer, wie in Abb. 4 zu sehen ist. Wenn eine Zutat eines Rezeptes aus unserem Golden Standard mehrfach erwähnt wird, ist es valide, dass sie dort nur einmal als solche ausgezeichnet ist. Dass unser Prototyp alle Vorkommen im Rezept auszeichnet, ist kein Fehler. Die Relevanz kann aber daher nicht mehr ausschließlich über die Wortposition verifiziert werden. Wenn die Verifizierung über die Wortposition fehlschlägt, versuchen wir es daher anschließend noch über die möglichen *ref*-Elemente. Beim Nachschlagen in unserem Zutaten-Wörterbuch speichern wir uns die möglichen *xml:ids*. Ist im Golden Standard Rezept eine der möglichen extrahierten *xml:ids* ausgezeichnet, wird die extrahierte Zutat ebenfalls als relevant evaluiert.

Abb. 4: Precision



Die Extraktion und Evaluation der ersten 50 Suppen dauert **knapp 2:30 Min.** Der **Recall** liegt bei **90%** (457 der 506 Zutaten wurden extrahiert) und die **Precision** ist **88%** (511 von 583 extrahierten Zutaten sind relevant). Diese Ergebnisse zeugen davon, dass die dictionary-based Extraktion funktioniert.

Version 0.2

Folgend testen wir nicht mehr nur die Machbarkeitsanalyse, sondern wollen eine echte Evaluierung durchführen. Daher machen wir uns zunächst genauere Gedanken, wie wir evaluieren, ob eine extrahierte Entity bezüglich einem manuell ausgezeichneten Rezept relevant und korrekt ist. Wie in Abb. 4 zu sehen ist, brauchen wir zwei unterschiedliche Vergleich-Ansätze, die wir **Exact match** und **Rough match** nennen wollen:

Ein **Exact match** kann nur vorliegen, wenn an gleicher Position im Rezept eine Zutat extrahiert wurde. Alle weiteren extrahierten Entities evaluieren wir anhand der Zutaten, da nur die Zutaten in cueML ausgezeichnet werden. Wenn beispielsweise korrekterweise 2 *EL* extrahiert wurde, aber es keiner Zutat zugeordnet werden kann, ist die extrahierte Information wertlos. Bei den Optionen 2 und 3 fordern wir für die Korrektheit nicht nur das Extrahieren der Zutat, sondern das zusätzlich das Extrahieren und Zuordnen von weiteren Entities wie beispielsweise eine Mengenangabe korrekt ist.

In der *Option 2* muss für einen exact match zusätzlich zu einer Zutat die Mengenangabe (das *quantity*-, *atLeast* und *atMost*-Attribut) und Einheit (*unit*-Attribut) korrekt extrahiert sein.

Option 3 überprüft zusätzlich die Attribute *optional* und *altGrp*. Die zu überprüfenden Attribute werden im Programm in *main.py* mit der Variabel *evalAttris* gesetzt.

Wenn bei der Precision kein Exact match vorliegt, folgt daraus nicht sofort, dass die Zutat nicht relevant ist, wie in Abb. 4 zu sehen ist. Wenn die extrahierte Zutat kein Exact match ist, aber an einer anderen Stelle im Rezept die gleiche Zutat extrahiert wurde und an der anderen Stelle ein Exact Match ist, liegt ein **Rough match** vor. Da diese extrahierte Zutat weder falsch ist, noch zu den relevanten Zutaten gehört, tun wir so, als ob wir diese Zutat nicht extrahiert hätten.

Tabelle 3 zeigt die Ergebnisse mit unserer finalen Version 0.2. Die Laufzeit beträgt unabhängig von den zu evaluierenden Optionen knapp **3 Min.**

Tabelle 3

	Recall	Precision
Option 1 (nur Zutaten)	99,23% (514 von 518)	98,10% (516 von 526)
Option 2 (mit Mengenangaben und Einheiten)	85,71% (444 von 518)	83,18% (445 von 535)
Option 3 (mit optionalen und alternativen Zutaten)	71,81% (372 von 518)	72,01% (373 von 518)

Der Recall und die Precision der Option 1 von fast 100% zeigen, dass nach den Verbesserungen in der Version 0.2 nahezu alle Zutaten extrahiert werden. Die verbliebenen Fehler sind hier aufgelistet.

Die Auswertung zu Option 2 zeigt, dass wir zu mehr als 8 von 10 Zutaten die Mengenangabe und Einheit richtig zuordnen. Wir haben dabei zwei systematische Fehler festgestellt. Zu einem können wir textuelle Mengenspannen der Form „auf jede Person $\frac{3}{4}$ Pfund, bei einer großen Gesellschaft $\frac{1}{2}$ Pfund“ nicht extrahieren. Stattdessen wird nur $\frac{1}{2}$ Pfund extrahiert. Zum anderen sind unsere vagen Mengenwörter wie *ein* oder *eine* problematisch. Wenn wir diese bei der Extraktion verwenden, wird zum Beispiel in „gibt man die Brühe [...] durch ein Sieb, schwitzt [...] Mehl in Butter [...]“ fälschlicherweise die Mengenangabe *ein/1* für Mehl extrahiert. Wenn wir solche Mengenwörter nicht berücksichtigen wird jedoch die Mengenangabe *1* beispielsweise bei „gibt man eine fein geschnittene Zwiebel hinzu“ nicht erkannt. In beiden Varianten gehen gut 20 Fehler auf die Verwendung bzw. nicht Verwendung solcher Mengenwörter zurück. Um die Mengenwörter von den unbestimmten Artikeln besser differenzieren zu können, werden offensichtlich weitere Regeln benötigt.

In Option 3 wird ersichtlich, dass unsere zwei beispielhaften Regeln für optionale und alternative Zutaten so gut wie gar nicht triggern. Dementsprechend gehen der Recall und die Precision in der Option 3 stark nach unten. Das Differenzieren dieser Entities durch Regeln muss definitiv noch weiter erforscht werden.

Dass die Anzahl unserer extrahierten Entities mit den Optionen variiert, liegt an unserem Rough match. In der Option 1 wird z. B. im Rezept B-1 das erste Vorkommen von Fleisch als Exact match identifiziert. Im Golden Standard Rezept ist nur das erste Vorkommen ausgezeichnet. Alle weiteren Vorkommen von Fleisch werden dann bei unserer Evaluation durch einen Rough match vom Ergebnis ignoriert. In der Option 2 wird jedoch beim ersten Vorkommen von Fleisch die Mengenangabe als falsch evaluiert. Daher schlägt anschließend auch der Rough match fehl und alle weiteren Vorkommen von Fleisch werden als Fehler identifiziert. In der Option 3 kommt es vor, dass Zutaten nun nicht mehr ein Exact match sind, da ihnen das *optional*-Attribut fehlt. Statt dessen werden sie jedoch durch einen Rough match gefangen. Daher tauchen in der Option 3 weniger Entities in der Precision auf. Der Rough match muss daher vielleicht für optionale und alternative Zutaten überdacht werden.

Insgesamt ist die Evaluation des dictionary- and rule-based Prototypen viel versprechend. Zukünftige Arbeiten sollten erforschen, inwieweit die Unterscheidung der Entities durch weitere Regeln verbessert werden kann sowie sich mit der Differenzierung von *ein/eine* als unbestimmter Artikel oder Zahlenwort beschäftigen.

Kommentare

Praktischer Nutzen der Arbeit	Hier werden ergänzend zur bisherigen Ausarbeitung Gedankengänge präsentiert, die wir während dieser Arbeit hatten.
Schwierigkeiten für die kulinarische Analyse mit cueML und Frau Davidis' Kochbuch	<h2>Praktischer Nutzen der Arbeit</h2> <p>Ich selber koche gerne. Beim Beschäftigen mit Auszeichnungssprachen wie auch Information Extraction habe ich darüber hinaus viel gelernt. Beides finde ich klasse, ist jedoch keine echte Rechtfertigung für ein Master-Arbeits-Thema. Folgend werden daher ein paar Beispiele vorgestellt, die durch (mit cueML) ausgezeichnete Rezepte ermöglicht werden.</p> <p>Wie bereits hier erwähnt, speichern die kommerziellen Seiten Chefkoch.de und Cooking.nytimes.com intern Meta-Daten zu ihren Rezepten. Dies zeugt von dem Bedarf für Auszeichnungssprachen in der Koch-Domäne. Nahe liegende, dadurch ermöglichte Services, sind Empfehlungssysteme für Rezepte, welche auf den Zutaten eines Rezeptes aufbauen. Das in (Ueda et al., 2011) vorgestellte System empfiehlt Rezepte, aufbauend auf den Zutaten der Rezepte, welche ein Nutzer in seiner Vergangenheit besucht hat. (Freyne, Berkovsky, 2010) und (Geleijnse et al., 2010) stellen Empfehlungssysteme vor, welche gesunde Rezepte bevorzugen. Wie gesund ein Rezept ist, wird aus den aufsummierten Nährwertangaben der verwendeten Zutaten bestimmt. (Teng et al., 2012) bietet darüber hinaus die Möglichkeiten nach Alternativen für eine Zutat eines Rezeptes zu suchen. So können beispielsweise einzelne Zutaten durch fettärmere ersetzt werden.</p>
Wissenschaftliche Arbeiten im 21. Jahrhundert	Aus Rezepten lassen sich auch gesellschaftswissenschaftliche Informationen ableiten. (Henning, 2008) beschäftigt sich zum Beispiel mit charakteristischen Merkmalen von Kochbüchern im Verlauf der deutschen Geschichte. So gibt es in Kochbüchern vor dem 18. Jahrhundert oft Hinweise, wie man den Eigengeruch von verdorbenen Zutaten überdecken kann, was ein Zeichen von Nahrungsmangel und Armut ist. Der prozentuale Anteil von Gerichten explizit für die christliche Fastenzeit zeugt von dem Einfluss der Religion. Die ersten Vorkommen von nicht lokalen Zutaten sind Indikatoren für internationalen Austausch. Ein anderes Beispiel ist (Ahn et al., 2011). Dort wird anhand von über 56.000 Rezepten die westliche mit der asiatischen Küche verglichen.
Wissen ist Macht	
Final Retrospective	

Schwierigkeiten für die kulinarische Analyse mit cueML und Frau Davidis' Kochbuch

Das Ziel dieser Arbeit ist es, eine kulinarische Analyse von Frau Davidis' Kochbuch vorzubereiten. Folgend gehen wir noch auf unbehandelte Aspekte ein. Dies beinhaltet zum einen Ungenauigkeiten im Kochbuch und zum anderen noch offene Punkte in cueML.

Nährwertangaben sind nur sinnvoll, wenn sie auf ein **Vergleichbares Maß** wie *pro Person* umgerechnet werden können. Ein Nudelsalat für 12 Personen wird insgesamt mehr Kalorien haben, als ein fetthaltiges 200g Steak. Bei Frau Davidis' Rezepten fehlen jedoch oft jegliche Angaben zu der angestrebten Portionen-Menge. Dementsprechend müsste diese geraten werden, was die Nährwertanalyse ungenau macht.

Gleiches gilt für die einzelnen Mengenangaben der Zutaten. Zum einen müssen vage Mengenangabe wie *ein Stich Butter* interpretiert werden. Meine Oma versteht z. B. sicherlich mehr als die fünffache Menge Butter darunter als meine Freundin. Dies führt zu weiteren Ungenauigkeiten bei der Nährwertanalyse. Zum anderen müssen historische Mengeneinheiten wie *ein Maß* oder *für 8 Pfennig Weißbrot* in heute übliche Einheiten umgerechnet werden, damit sie vergleichbar werden. Dafür ist historische Recherche nötig. Auch für manche Zutaten wie *ganzes Gewürz* sind weitere Recherchen nötig, um herauszufinden, was damit gemeint ist.

Weitere Schwierigkeiten bereitet die **schlechte Struktur von Frau Davidis' Kochbuch**. Manche Informationen sind in anderen Rezepten versteckt. Im Rezept *Rindfleischsuppe mit Perlgerste und Reis* steht z. B. „Anmerk. Will man Reis oder Sago zur Suppe nehmen, so gibt man dieses später hinein. Man rechnet davon auf jede Person bei allen Fleischsuppen einen gestrichenen Eßlöffel voll“ (Davidis, 1849, S. 29). Zu allen Fleischsuppen gehört also als optionale Zutat pro Person ein Eßlöffel Reis oder Sago, was man nur wissen kann, wenn man dieses Rezept gelesen hat.

Überhaupt das Vorkommen von optionalen Zutaten erschwert die kulinarische Analyse. Je nachdem, ob eine optionale Zutat verwendet wird, hat ein Rezept eine andere Geschmacksrichtung und natürlich auch andere Nährwertangaben. Den Mittelwert der Nährwertangaben vom Verwenden und nicht Verwenden einer optionalen Zutat zu bilden, macht kulinarisch keinen Sinn. Gleiches gilt für alternative Zutaten. Das Rezept für *Barsch auf deutsche Art* (Davidis, 1849, S. 62) gibt entweder als Soße geschmolzene Butter mit Senf oder eine Eier-Sauce hinzu. Je nachdem welche Sauce verwendet wird, hat das Rezept ein anderes Flair. Streng genommen ist jede alternative Zutat wie auch jede optionale Zutat ein eigenes Rezept. Ist in einem Rezept eine Zutat optional und sind drei weitere alternativ zueinander, enthält das Rezept eigentlich sechs Rezepte. Zwei je nachdem, ob die optionale Zutat verwendet wird und jeweils drei dazu, je nachdem welche der alternativen Zutaten verwendet wird. Eine kulinarische Analyse muss daher aus den Rezepten die *echte* Anzahl der Rezepte extrahieren.

Die Anbindung von cueML an den **Bundeslebensmittelschlüssel** (BLS) erfüllt die Anforderung der frei verfügbaren Ressource nicht. Ich sehe ihn allgemein aus folgenden Gründen kritisch:

- Er ist nicht öffentlich zugänglich (wir dürfen ihn unter einer Forschungslizenz verwenden, jedoch in keiner Weise veröffentlichen). Damit sind durch ihn berechnete Nährwertangaben nicht nachvollziehbar. Somit geht die Transparenz verloren, die ein wichtiger Bestandteil des semantischen Webs ist (Berners-Lee et al., 2001).
- Wie die Nährwertangaben einzelner Zutaten berechnet werden, ist bereits im BLS nicht transparent. Nach dem BLS haben z. B. *Nichtalkoholische Getränke* einen Energiegehalt von 47kcal/100g, 198kJ/100g, ... Da frage ich mich, wieso manche Leute aus Diät-Gründen auf Cola verzichten, wenn Wasser wie Cola als nicht alkoholisches Getränk doch die gleichen Nährwertangaben hat. Als anderes Beispiel steht in einer Liste über derzeit bekannte Diskrepanzen in BLS - Version 3.02: „Der Fettgehalt von U661100 „Schwein Schulter (Bug) (ma) roh“ hat sich von Version II.3

zu 3.02 fast verdoppelt. Ursache: Bereits im Zuge des Updates für Version 3.0 wurde gegenüber Version II.3 ein aktuellerer Analysenwert übernommen. **WICHTIG:** Dabei handelt es sich nicht um einen Fehler.“ Die Korrektur einer Angabe um fast 100% ist nicht sehr vertrauenerweckend, wenn als Begründung lediglich pauschal ein aktuellerer Analysewert angegeben wird.

- Ein Mapping Zwischen einer Zutat und dem BLS kann sich als überraschend schwierig erweisen. Zu *Hauskaninchen* hat der BLS z. B. 17 Einträge (Hauskaninchen gegart, Hauskaninchen roh, Hauskaninchen gekocht, ...). Dem liegt das allgemeine Problem zugrunde, dass es nicht die perfekte Muster-Zutat gibt. Streng genommen hat auch jedes individuelles Hauskaninchen (roh, gekocht, ...) individuelle Nährwertangaben. Eine kulinarische Analyse basierend auf Nährwertangaben kann im Allgemeinen nicht exakt sein, da selbst zwei benachbart gewachsene Möhren aus dem selben Feld unterschiedliche Nährwertangaben haben. Eine Verallgemeinerung erhöht die Übersichtlichkeit und somit auch die Nutzbarkeit und Vergleichbarkeit.
- Die Struktur des BLS-Schlüssels lässt zwar eine Taxonomy erahnen, allerdings ist diese nicht konsequent eingehalten und auch nicht dokumentiert.

Da wir cueML entwickelt haben, sind wir noch die einzigen, die dieses Vokabular verwenden. Idealerweise würden nach einer Veröffentlichung **sämtliche Kochseiten cueML adoptieren**. Das würde die in Praktischer Nutzen der Arbeit erwähnten Anwendungen ermöglichen.

Diese Arbeit hat sich für den Prototypen zur automatischen Auszeichnung erst einmal nur mit den ersten 50 Rezepten beschäftigt. Das Wörterbuch aller Zutaten konnte aus den manuell ausgezeichneten Rezepten extrahiert werden. Um neue Rezepte auszuzeichnen wird ein **allgemeines Wörterbuch von Zutaten** benötigt. Wie zuvor erwähnt, ist der BLS dafür nicht ideal. Als Alternative haben wir auch GermaNet in Betracht gezogen. In diesem Thesaurus gibt es *Nahrung* als Unterkategorie von Nomen. Allerdings ist GermaNet auch nicht frei verfügbar. Des Weiteren sind zu den einzelnen Begriffen keine Nährwertangaben angegeben. Einträge wie *Fressen*, *Mahlerzeugnis*, *Gang*, *Biokleidung*, *Dübel*, *Haschzigarette*, usw. lassen außerdem auch Zweifel an der Qualität dieser Unterkategorie aufkommen. Ich habe den allgemeinen Verdacht, dass frei zugängliche Datenquellen nicht nur nachvollziehbar sind, sondern durch die Transparenz und der Teilung mit vielen Interessierten auch qualitativ besser sind.

Wissenschaftliches Arbeiten im 21. Jahrhundert

Als ich 2014 meine Bachelor-Arbeit in *Word* geschrieben habe, wurde ich von allen gefragt, ob ich noch bei klarem Verstand bin - eine wissenschaftliche Arbeit ist doch selbstverständlich in *LaTeX* zu schreiben. Dem widerspreche ich vehement. „Vor meiner Zeit“ mag *LaTeX* das mit Abstand beste und stabilste Textsatzsystem gewesen sein, so dass größere Arbeiten nur mit diesem sinnvoll zu schreiben waren. Inzwischen wurde *Word* hingegen seit mehr als 34 Jahre weiter entwickelt und Funktionen wie Bibliographien, Inhaltsverzeichnisse, Formeln, usw. funktionieren in *Word* problemlos.

Wie anhand dieser Webseite zu sehen ist, will ich jedoch auch nicht *Word* als das Mittel der Wahl verkaufen. Wir denken, dass Papier im Allgemeinen für Abschluss-Arbeiten nicht mehr das ideale Medium ist. Gebundene exemplare tendieren dazu im Regal einzustauben und nie mehr gefunden zu werden. Sie können auch schlecht weiterentwickelt werden.

Bereits 1989 hat Tim Berners-Lee in seinem Proposal am CERN, welches als Grundstein für das World Wide Web gilt, geschrieben, dass Dokumentation durch ein Buch zu einem System, welches ständig von unterschiedlichen Leuten verwendet und weiterentwickelt wird, unbrauchbar ist (Berners-Lee, 1989). Eine Master-Arbeit verstehen wir als so ein, zeitlich begrenztes Forschungsprojekt, welches idealerweise in weiteren Forschungsarbeiten von anderen Leuten als dem nun fertig absolvierten Studenten weiterentwickelt wird.

Wie diese Arbeit zeigt, kann eine Abschluss-Arbeit auch als Webseite dargeboten werden. Abbildungen profitieren vom dem potenziell größeren Bildschirm als das feste A-4 Format. Auch sind interaktive Grafiken möglich. Des Weiteren können Webseiten zu Hause am PC, auf der Couch am Tablet oder unterwegs am Handy gelesen werden. Links auf einer Webseite zu folgen ist angenehmer, als sie über ein gedrucktes Literaturverzeichnis manuell aufzufinden. Über die URL ist die Arbeit leicht verteilbar und durch das World Wide Web frei zugänglich und auffindbar. Durch eine schicke Webseite kann ein Thema auch (teilweise) dem Laien zugänglich gemacht werden.

Ein weiterer Vorteil bietet die Integration eines Blogs in die Webseite. Seine Gedanken niederzuschreiben ist immer sinnvoll. Dies hilft die Gedanken zu ordnen und früh zu merken, wo sie noch nicht stimmen. Der Blog bietet auch eine gute Plattform zum Austausch von Informationen. Der betreuende Professor ist nach dem Lesen stets im Bilde, was der Student erarbeitet hat. Unter die Blog-Einträge kann er dann ggf. sofort Feedback geben. So ist ein aneinander vorbeireden unwahrscheinlich und die stattgefundene Kommunikation ist später noch nachvollziehbar. Über den Blog können auch andere Wissenschaftler Feedback geben und ihre Erkenntnisse teilen.

Meiner Meinung nach hätten mich also bereits 2014 alle fragen sollen **wieso schreibst du keine Webseite für deine Abschluss-Arbeit?** Unten stehend noch eine schöne Karikatur von einer schicken, aufgeräumten und vernetzten Webseite vs. einer verstaubten, gebundenen Abschluss-Arbeit.

*Gemalt von Fenja Severing nach Idee von Torsten Knauf



Wissen ist Macht

Mit Forschung ist auch immer die Frage der Verantwortung verbunden. Ein einprägsames Beispiel dafür ist die Frage, inwieweit Physiker, die an der Atom-Bombe geforscht haben, Verantwortung für den Einsatz einer Atom-Bombe tragen. Eine sehr empfehlenswerte Lektüre zu diesem Thema ist „Die Physiker“ von Friedrich Dürrenmatt. Unsere Arbeit beschäftigt sich damit, wie Wissen automatisiert extrahiert und ausgezeichnet werden kann. Daher möchte ich folgend erörtern, dass nicht nur technische Forschungen wie an der Atom-Bombe oder die Erfindung des Internets eine Gesellschaft verändern können sondern auch Wissen.

Bereits im 7.Jh. wird dem Schwiegersohn des Propheten Mohammeds das Zitat „*Knowledge is power...*“ zugeordnet. Aber „*With great power comes great responsibility*“. Dies möchte ich folgend an ein paar Beispielen verdeutlichen.

Nahj Al-Balagha, Saying 146 (Imam Ali, ca. 599-661 n.Chr.)

„Knowledge is power and it can command obedience. A man of knowledge during his lifetime can make people obey and follow him and he is praised and venerated after his death. Remember that knowledge is a ruler and wealth is its subject“

Bekanntes Sprichwort mit unbekannten Ursprung (s. hier für genauere Informationen)

„With Great Power Comes Great Responsibility“

Das für mich persönlich krasseste Beispiel, wie Wissen instrumentalisiert und missbraucht werden kann, ist die Nazi-Zeit unter Adolf Hitler. Ein wesentlicher Bestandteil der Unrechts-Diktatur war sicherlich falsche Anschuldigungen gegenüber Juden, um ein gemeinsames Feindbild zu haben. Ein anderer die Hitlerjugend welche früh der Jugend eine Ideologie und Weltanschauung eingetrichtert hat. Beides sind Formen von Wissen.

Ein sehr aktuelles Beispiel ist die Diskussion, in wie weit *Fake News* öffentliche Meinungen und Wahlen beeinflussen (s. z. B. hier). Ich persönlich denke, dass die Methodik der bewusst falschen Informationen zum eigenen Vorteil kein neues Phänomen ist. Ich behaupte, dass es bereits seit dem Bestehen von Zeitungen Zeitungsenten gibt, um die Verkaufszahlen der Zeitung in die Höhe zu treiben. In den beiden Weltkriegen hat man kritische wie auch ausländische Presse oft als *Lügenpresse* angeprangert (ein Begriff der traurigerweise aktuell wieder im Umlauf ist). Die Amerikaner sind 2003 in den Irak einmarschiert mit der Begründung, dass der damalige Diktator Saddam Hussein Massenvernichtungswaffen besitzt, was sich im Nachhinein als eine Lüge herausgestellt hat. Erst kürzlich wurden die im Baltikum stationierten Bundeswehrtruppen fälschlicherweise einer Vergewaltigung beschuldigt, um die Toleranz ihnen gegenüber zu vermindern (s. z. B. hier). All dies verdeutlicht die Macht von Informationen / Nachrichten / (falschem) Wissen. Was meiner Meinung nach neu ist, ist dass durch das Internet quasi jeder, anonym und ohne Zeitverzögerung ein großes Publikum erreichen (und auch belügen) kann.

Ein weiteres Beispiel ist die bloße Existenz von Geheimdiensten, zu deren Hauptaufgabe es gehört, Informationen zu beschaffen. So war z. B. in den Medien, dass der russische Geheimdienst Material gesammelt hat, um den aktuellen amerikanischen Präsidenten Donald Trump zu erpressen (s. z. B. hier). „*Knowledge is power and it can command obedience*“ lässt grüßen. Seit 2002 bis vielleicht sogar heute, wird auch von den amerikanischen Geheimdiensten das Handy des deutschen Bundeskanzlers abgehört (s.

z. B. hier). Nur ein Schelm (wie ich) würde behaupten, dass das etwas mit *mangelnden command obedience* von dem damaligen Bundeskanzler Gerhard Schröder, den Amerikanern in den Irak-Krieg zu folgen, zu tun haben könnte.

In der Koch-Domäne die Macht des Wissens zu diskutieren, mag vielleicht übertrieben sein. Ich bin jedoch der Meinung, dass es nie falsch ist, das Thema zu diskutieren. Daher möchte ich diese kleine Abschweifung mit einem kurzen Gedankenspiel beenden. Dieses soll zeigen, wie selbst Wissen, welches durch unser cueML-Vokabular zugreifbar geworden ist, Macht ausüben könnte. Stellen Sie sich vor, es sind Wahlen und Sie wollen natürlich zukünftig einen gesunden BundeskanzlerIn haben. Nun lesen Sie in der Zeitung, dass jemand den Internet-Verkehr ihres Kandidaten mitgeschnitten hat. Seit zwei Monaten öffnet er verstärkt Rezepte, die diese X Nährstoffe meiden. Wie Ihnen jeder Arzt bestätigen kann, macht es Sinn diese Nährstoffe zu meiden, wenn man Bauchspeicheldrüsenkrebs hat. Der Artikel kommt also zu dem Ergebnis, dass der Kandidat schwer krank ist. Der Kandidat selber weigert sich jedoch aus Prinzip, seine Kranken-Akte wegen einer so lächerlichen Anschuldigung öffentlich zu machen, da damit viele private Informationen verbunden sind. Denken Sie, dass dieses Szenario ihre Wahlentscheidung oder die von anderen beeinflussen würde? Der Gesundheitszustand von Hillary Clinton war auf jeden Fall ein Thema im amerikanischen Wahlkampf 2016, aufgebauscht durch einen kleinen Schwächeanfall von ihr und ihrer Weigerung ihre Kranken-Akte zu veröffentlichen (s. z. B. hier).

Final Retrospective

Abschließend möchte ich Revue passieren lassen, was ich in dieser Arbeit gelernt habe. Ich denke bei jedem Software Projekt sollte man etwas lernen. Dies gilt insbesondere für eine Abschluss-Arbeit. Dieses Revue orientiert sich dabei an folgenden vier Fragen aus (Kerth, 2001): 1. *What did we do well, that if we don't discuss we might forget?* 2. *What did we learn?* 3. *What should we do differently next time?* 4. *What still puzzles us?*

1. Zuerst *quick and dirty* Prototypen als Machbarkeitsanalyse zu erstellen war eine gute Idee. So wurden früh Probleme erkannt und der CRF-based Ansatz nicht weiter verfolgt. Den dictionary- and rule-based Prototypen vor der Weiterentwicklung neu zu designen war ebenfalls sehr wichtig. Die anschließende Integration der rule-based Weiterverarbeitung, sowie das zukünftige Hinzufügen von weiteren Regeln, ist auf Grund des guten Designs trivial. Die Unit-Tests zeitgleich mit dem Source Code zu schreiben hat auch gut funktioniert. Dies war noch eine wichtige Lektion aus meiner Bachelor-Arbeit.
2. Natürlich habe ich viele Kleinigkeiten davon gelernt, interessante Artikel zu lesen und ein größeres Projekt vom Anfang bis zum Ende durchzuführen. Mein einschlagendes *Aha-Erlebnis* hatte ich jedoch sicher bezüglich Domänen-spezifischen Vokabulare. Ich habe stets gedacht, dass sich jemand ein Vokabular trivial aus den Fingern saugt und es dann von vielen Leuten verwendet wird. Aber Vokabulare aufzubauen ist zeitaufwendig, benötigt Fachwissen und ist daher auch teuer. Auch ist es nicht einfach ein passendes Vokabular zu finden. Ein universelles Vokabular für alle Zwecke ist illusorisch. Diese (für mich neue Erkenntnisse) werden u.A. in (Zeeshan, 2009) und (Hage et al., 2008) bestätigt.
3. Folgende zwei Sachen würde ich verbessern, wenn ich die Möglichkeit hätte, noch einmal von vorne anzufangen:
 - Im Rahmen meiner SHK-Tätigkeit habe ich viele Rezepte aus Davidis' Kochbuch manuell ausgezeichnet. Dies war gut, da ich dadurch ein Gefühl

dafür bekommen habe, wie schwierig das Auszeichnen der Rezepte ist. Allerdings habe ich sehr viele Rezepte ausgezeichnet, bevor ich *Stopp* gesagt habe - *Wir müssen unser cueML Vokabular ändern*. Dann habe ich wieder sehr viele Rezepte mit dem geänderten Vokabular ausgezeichnet, bevor ich erneut *Stopp* gesagt habe - *Das stimmt noch immer nicht*. Stattdessen hätte ich mich an das Prinzip aus dem Agilen Manifest *Working software is the primary measure of progress* halten sollen. Das cueML Vokabular alleine ist keine funktionierende Anwendung. Der Sinn des Vokabulares ist es, Informationen so auszuzeichnen, dass sie von einer Anwendung verwendet werden können. Dementsprechend hätte ich nach einer Änderung im Vokabular den Workflow bis zur Webseite einmal exerzieren sollen; ein paar Rezepte auszeichnen, diese Rezepte in eine Webseite umwandeln und mir dann als Nutzer anschauen, ob ich die Rezepte passend dargestellt finde. Daraufhin kann das Vokabular erneut angepasst werden, usw. So hätte verhindert werden können, dass ein Großteil meines Auszeichnens als SHK letztendlich nicht verwendet wird, da es mit einer alten Version des cueML-Vokabulares ausgezeichnet ist.

Intuitiv stellt sich die Frage, ob dies auch durch eine *Requirement Analyse* hätte verhindert werden können. Alle Anforderungen wie optionale Zutaten, oder Verweise zwischen (Teil)-Rezepten *upfront* zu erkennen, halte ich jedoch für illusorisch. Dementsprechend ist der oben beschriebene *iterative* Ansatz besser.

- Des Weiteren würde ich, wie in Wissenschaftliches Arbeiten im 21. Jahrhundert erläutert, von Anfang an einen echten Blog führen.
- 4. Beim nächsten größeren Projekt werde ich versuchen meine Anfangserwartungen herunterzuschrauben. Am Anfang habe ich gedacht, dass ich selbstverständlich alle Informationen aus Davidis' Kochbuch extrahieren kann. Nach und nach musste ich jedoch erkennen, dass das gelinde gesagt übermotiviert war. Unser finaler vorgestellte Prototyp ist davon noch weit entfernt, was ihn in keiner Weise schlecht reden soll.

Zusammenfassung & Ausblick

Zusammenfassung

Ausblick

Zusammenfassung

Das Ziel dieser Arbeit ist es, Frau Davidis' Kochbuch für eine kulinarische Analyse digital aufzubereiten. Die digitale Edition der Rezepte ist hier zu finden.

Bis zu dieser Arbeit gab es noch kein frei verfügbares Vokabular in der Koch-Domäne, welches eine kulinarische Analyse von Rezepten ermöglicht. Daher haben wir Anforderungen an so ein Vokabular aufgestellt und darauf aufbauend *culinary editions Markup Language (cueML)* entwickelt. CueML erweitert den Standard der Text Encoding Initiative (TEI) und Schema.org/Recipe. Es führt unter anderem Möglichkeiten zur Auszeichnung von Mengenangaben und Einheiten von Zutaten ein sowie Unterscheidungen zwischen optionalen und alternativen Zutaten. Sowohl Verweise auf Teil-Rezepte als auch allgemeine Verweise auf andere Stellen können gesetzt werden.

Da das manuelle Auszeichnen zeitaufwendig und fehleranfällig ist, haben wir des Weiteren an Prototypen zum automatischen Auszeichnen geforscht. Dazu müssen die auszuzeichnenden Entities zuerst extrahiert werden. Für das Extrahieren haben wir zwei verschiedene Ansätze ausprobiert. Der Conditional Random Field-based Prototyp hat sich nicht als zielführend erwiesen. Der dictionary- and rule-based Prototyp ist hingegen vielversprechend. Dieser lemmatisiert zu erst jedes Wort des Textes. Die Lemmatisierung baut auf TreeTagger auf. Aufgrund der alten verwendeten deutschen Sprache von Frau Davidis entwickeln wir eine einfache Überarbeitung des Outputs von TreeTagger. Nach der Lemmatisierung können die Lemmata in Wörterbüchern nachgeschaut werden, welche alle gesuchten Entities enthalten. In der Koch-Domäne ist die Annahme der Existenz solcher Wörterbücher vertretbar. Nachdem die Entities extrahiert wurden, können diese mittels Regel weiterverarbeitet werden. Die *rule-based* Weiterverarbeitung ermöglicht z. B. das Zuordnen von Mengenangaben zu Zutaten oder eine Differenzierung, dass eine Zutat im Rezept optional ist.

Ausblick

Diese Arbeit war der Startschuss in ein neues Forschungsfeld. Dementsprechend stehen noch weitere Arbeiten an, die wir folgend kurz auflisten wollen:

- Die automatische Extraktion beschränkt sich zur Zeit noch auf Zutaten, sowie ihre Mengenangaben und Einheiten. Es fehlen noch folgende Entities: Vorbereitungszeit, Kochzeit, gesamte Zubereitungszeit, Zubereitungsmethode, Verweise auf allgemeine Stellen und Verweise auf inkludierte Teil-Rezepte.
- Eine weitere Differenzierung von Zutaten in *Beilagen* erscheint uns sinnvoll. Ansonsten würden z. B. beim Rezept „*Schmalz- oder Butterkohl*“ aufgrund von „*Beilagen: Schinken, Rauchfleisch, Bratwurst.*“ Bratwurst als Zutat in der Zutatenliste auftauchen.

- Verbesserung bzw. hinzufügen von weiteren Regeln in der rule-based Weiterverarbeitung des dictionary- and rule-based Prototypen. Um die durch Regeln extrahierten Informationen nachvollziehen zu können, falls beispielsweise fehlerhafte Informationen extrahiert wurden, sollte die Historie von angewendeten Regeln transparent sein. Diese sollten nicht in der Auszeichnungssprache cueML integriert werden, sondern als Debugging-Informationen dem Software-Entwickler verfügbar gemacht werden.
- Das Finden bzw. das Aufbauen eines allgemeinen Zutaten-Wörterbuches; idealerweise als weltweit frei verfügbare, eindeutige Ressource, mit Nährwertangaben.
- Veröffentlichen und verbreiten von cueML.
- Und zu guter Letzt natürlich die eigentliche kulinarische Analyse.

Quellen

- Skip The Pizza auf WordPress.org (2012).** "Skip The Pizza". <http://skipthepizza.com/blog/analyzing-the-ingredients-of-29200-recipes>. Abgerufen am 2016-10-28.
- Feldman, Ronen; Dagan, Ido (1995).** "Knowledge Discovery in Textual Databases (KDT)". In: Proceedings of the First International Conference on Knowledge Discovery and Data Mining. S. 112-117. Montreal, Kanada. AAAI Press. <http://dl.acm.org/citation.cfm?id=3001335.3001354>.
- Ahn, Yong-Yeol; Ahnert, Sebastian E.; Bagrow, James P.; Barabási, Albert-László (2011).** "Flavor network and the principles of food pairing". <http://www.nature.com/articles/srep00196>. Abgerufen am 2017-01-14.
- Davidis, Henriette (1849).** "Praktisches Kochbuch für die gewöhnliche und feinere Küche, 4. vermehrte u. verbesserte Auflage". Bielefeld. Velhagen und Klasing.
- Deutsches Textarchiv (A) (2008).** "Davidis, Henriette: Praktisches Kochbuch für die gewöhnliche und feinere Küche. 4. Aufl. Bielefeld, 1849 - Digitalisiert vom Deutschem Textarchiv". Berlin-Brandenburgische Akademie der Wissenschaften. http://www.deutschestextarchiv.de/book/show/davidis_kochbuch_1849. Abgerufen am 2017-01-06.
- Deutsches Textarchiv (B) (2017).** "DTA - Projektüberblick". Berlin-Brandenburgische Akademie der Wissenschaften. <http://www.deutschestextarchiv.de/doku/ueberblick>. Abgerufen am 2017-01-07.
- Erdmann, M.; Maedche, A.; Schnurr, H.-P.; Staab, S. (2001).** "From Manual to Semi-automatic Semantic Annotation: About Ontology-based Text Annotation Tools".
- Berners-Lee, Tim (1989).** "Information Management: A Proposal". <https://www.w3.org/History/1989/proposal.html>.
- 06onkel von Chefkoch.de (2017).** "Rezept für Pfannekuchen". <http://www.chefkoch.de/rezepte/363861121870267/Pfannekuchen.html>. Abgerufen am 2017-01-07.
- Berners-Lee, Tim; Hendler, James; Lassila, Ora (2001).** "The Semantic Web". In: Scientific American, 2001-05, S. 29-37.
- Ueda, Mayumi; Takahata, Mari; Nakajima, Shinsuke (2011).** "User's Food Preference Extraction for Personalized Cooking Recipe Recommendation". <http://dl.acm.org/citation.cfm?id=2887675.2887686>.
- Greene, Erica (2015).** "Extracting Structured Data From Recipes Using Conditional Random Fields". In: The New York Times. https://open.blogs.nytimes.com/2015/04/09/extracting-structured-data-from-recipes-using-conditional-random-fields/?_r=1. Abgerufen am 2017-01-08.
- Geleijnse, Gijs; Overbeek, Thérèse; Veeken, Nick van der; Willemsen, Martijn (2010).** "Extracting Vegetable Information from Recipes to Facilitate Health-Aware Choices".
- Freyne, Jill; Berkovsky, Shlomo (2010).** "Recommending Food: Reasoning on Recipes and Ingredients". http://dx.doi.org/10.1007/978-3-642-13470-8_36.
- Hohto, Andreas; Nürnberger, Andreas; Paaß, Gerhard (2005).** "A brief survey of text mining". In: LDV Forum - GLDV Journal for Computational Linguistics and Language Technology.
- Imam Ali (ca. 599-661 n.Chr.).** "Nahj Al-Balagha, Saying 146".
- Hage, Willem Robert van; Sini, Margherita; Finch, Lori; Kolb, Hap; Schreiber, Guus (2008).** "The OAEI food task: an analysis of a thesaurus alignment task".
- Henning, Beate (2008).** "Daz ist ein guot geriht und versaltz ez niht". In: Texte für Technik - Ausgabe Herbst 2008. S. 4-5.
- Kashyap, Vipul; Bussler, Christoph; Moran, Matthew (2008).** "The Semantic Web". Springer-Verlag Berlin Heidelberg.
- Kerth, Norman L. (2001).** "Project Retrospectives: A Handbook for Team Reviews".
- Schema.org (2017).** "<http://schema.org/docs/datamodel.html>". <http://schema.org/docs/datamodel.html>. Abgerufen am 2017-01-09.
- Sutton, Charles; McCallum, Andrew (2012).** "An Introduction to Conditional Random Fields". <http://dx.doi.org/10.1561/22000000013>.
- TEI-Konsortium (A) (2017).** "TEI-Konsortium". University of Virginia Library, Charlottesville VA 22904-4114, USA. <http://www.tei-c.org/index.xml>.
- TEI-Konsortium (B) (2017).** "TEI P5: Guidelines for Electronic Text Encoding and Interchange. Version 3.1.0.". <http://www.tei-c.org/Guidelines/P5/>. Abgerufen am 2017-01-07.
- Teng, Chun-Yuen; Lin, Yu-Ru; Adamic, Lada A. (2012).** "Recipe Recommendation Using Ingredient Networks". <http://doi.acm.org/10.1145/2380718.2380757>.
- Wiegand, Michael; Roth, Benjamin; Klakow, Dietrich (2012).** "Data-driven Knowledge Extraction for the Food Domain". Saarbrücken, Deutschland.
- Zeeshan, Ahmed (2009).** "Domain Specific Information Extraction for Semantic Annotation". Diplom-Arbeit an der Charles University in Prague, Tschechien und der University of Nancy in Frankreich.