Evaluierung

Auf dieser Seite werden die zuvor vorgestellten Prototypen (AutomatischesAnreichernCueML.html) 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) (Quellen.html#TextMining) 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.

$$Recall = \frac{\#(retrieved \cap relevant)}{\#relevant} \tag{1}$$

$$Precision = \frac{\#(retrieved \cap relevant)}{\#retrieved} \tag{2}$$

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 (AutomatischesAuszeichnenCueML.html##CRFPrototyp)

Antrainiert haben wir den CRF-based Prototypen mit diesen 9 Rezepten (images/AutomatischesAnreichrenCueML/Trainings-Rezepte.txt). Unser Test-Rezept (images/AutomatischesAnreichrenCueML/Test-Rezept.txt) 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 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 Entitites des Rezeptes nicht erkannt sondern als *O* gelabelt.

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%

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 ausgezeichneter 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) (Quellen.html#CRFNYT) 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) (Quellen.html#CRFNYT**)**.
- 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 Rezepte gibt**. Aus den 10 verwendeten Rezepten ergeben sich mehr als 1.500 Zeilen, die gelabelt werden müssen. Wie (Greene, 2015) (Quellen.html#CRFNYT) 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) (Quellen.html#CRFNYT) 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.

Abb. 2: Ausschnitt unserer CRF Trainingsdaten

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

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) (Quellen.html#CRFNYT) 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

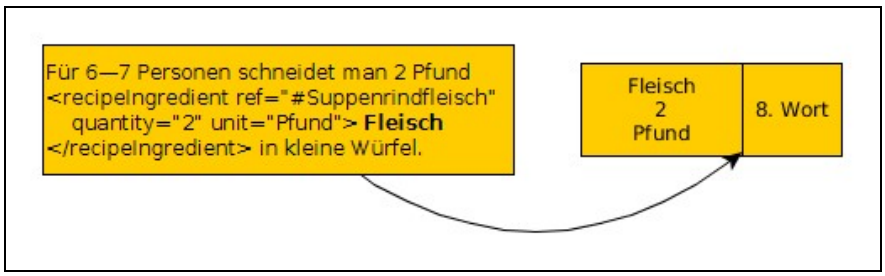
Unsere dictionary- and rule-based Prototypen testen wir anhand von Recall und Precision. Wir haben dazu die ersten 50 Suppen von Frau Davidis' Kochbuch (DavidisesKochbuch/Rezepte%20mit%20cueML.xml) manuell ausgezeichnet und verwenden dies als Golden Standard. Unsere Prototypen zeichnen die gleichen 50 Rezepte aus und werden dann gegen unseren Golden Standard evaluiert.

Version 0.1 (AutomatischesAuszeichnenCueML.html#DictBasedV01)

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.

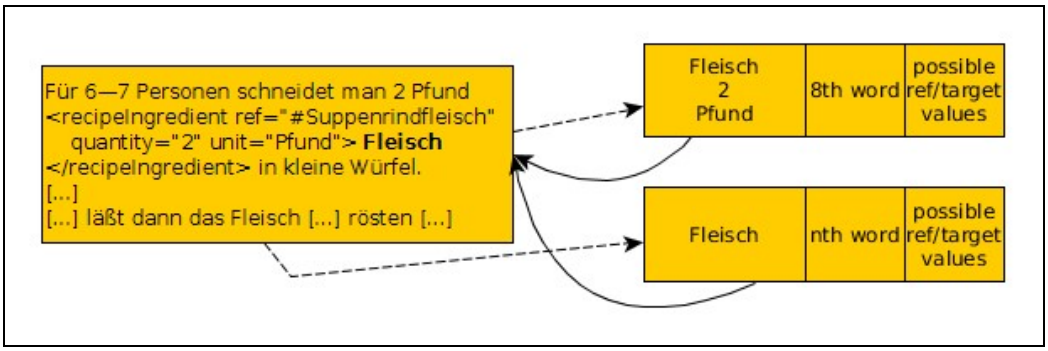
Abb. 3 zeigt, wie wir den **Recall** ausgewertet haben. Idealerweise ist in den extrahierten Zutaten jede Zutat wiederzufinden, die in unserem Golden Standard auch manuell ausgezeichnet wurde. Ist ein Wort im Golden Standard als Zutat ausgezeichnet, überprüfen wir einfach, ob unser Prototyp an der gleichen 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 (AutomatischesAuszeichnenCueML.html#DictBasedV02)

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 Entitiy 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 *quantitiy*-, *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 (verbliebeneFehler.html) aufgelistet.

Die Auswertun 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 ¾ Pfund, bei einer großen Gesellschaft ½ Pfund“* nicht extrahieren. Stattdessen wird nur *½ 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 unser 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.