

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

Abstract

Always do this one last, when knowing the things to praise yourself for :P

Acknowledgements

If I don't profit from nice people in these thesis, I have done something horrible wrong. So try to remember most of them here at the end... :)

Contents

1	Introduction	1
2	Making a cookbook machine readable	2
2.1	Digitalisation	2
2.2	CueML ontology	2
2.3	Applying cueML to our recipes	5
2.4	Need for automation	6
3	Related Work	7
3.1	Precision & Recall	7
3.2	Skip The Pizza	7
3.3	Extracting Structured Data From Recipes Using Conditional Random Fields	8
3.3.1	Conditional Random Fields	8
3.3.2	Implementation of New York Times	10
3.4	Domain Specific Information Extraction for Semantic Annotation . .	12
3.5	Distinction to this work	13
4	Development of recipe parser	14
4.1	Starting position	14
4.2	Overall picture	14
4.2.1	Workflow	14
4.2.2	Features	14
4.2.3	Evaluation	14
4.3	First Iteration: Basis CRF	14
4.3.1	Workflow	14
4.3.2	Features	14
4.3.3	Evaluation	14
4.4	Final recipe parser	14
4.4.1	Workflow	14
4.4.2	Features	14
4.4.3	Evaluation	14
5	Discussion	15
5.1	Power of machine readable data	15
6	Summary	16
A	Statutory Declaration	17

B Full example use of cueML	18
C RELAX NG grammar for cueML	21

List of Figures

2.1	A recipe from our cookbook	3
2.2	Schema.org/Recipe example from (Schema.org 2016)	4
3.1	John likes apple (Ahmed 2009)	13

List of Tables

3.1	Evaluation Domain Specific Information Extraction for Semantic An- notation	13
-----	--	----

Chapter 1

Introduction

A recipe parser, which can tag old and rather unstructured cookbooks according to the ontology of (Schema.org 2016), is developed in this thesis. Once it is tagged, it is easy to extract the tagged entities. The parser is developed and tested with *Davidis, Henriette: Praktisches Kochbuch für die gewöhnliche und feinere Küche. 4. Aufl. Bielefeld, 1849* but can be adapted easily to every German cookbook or website. For other languages new training data and dictionaries for the machine learning preparation of the parser have to be provided, but the general algorithm can be inherited.

This effort is motivated by nutritional science. Being able to extract the ingredients of a recipe automatically simplifies research according healthy food. But also sociological analysis are enabled through that.

The thesis is structured as follows:

Chapter 2

Making a cookbook machine readable

This chapter covers shortly, how we transform a cookbook in a machine readable format, which can be arbitrarily processed further. To achieve this, the cookbook has to be digitalised first. Afterwards it has to be enriched with meta data from an ontology.

2.1 Digitalisation

In general there are two different ways, how to digitalise a book. The first one is to scan each side and let an optical character recognition program extract the text of the scanned pictures. The second one is to type it manually into a computer.

The German Text Archive provides a collection of German texts from the 16th to the 19th century including our targeted cookbook *Davidis, Henriette: Praktisches Kochbuch für die gewöhnliche und feinere Küche. 4. Aufl. Bielefeld, 1849* in (Deutsches Textarchiv 2016). They digitalised it through double keying, meaning that two people independent of each other manually typed the book into the computer. Differences in their versions were revised by a third person. They have already enriched the book with *TEI: Text Encoding Initiative*-standard¹. TEI is a standard for representing printed text in digital form. As many as possible characteristics of the printed medium are kept through meta data. Its main purpose is for analysing in humanities, social sciences and linguistics.

Because we are only interested in extracting certain data from the recipes and not in linguistic analysis or something else, we have transformed the digitalised version as depicted in fig. 2.1 on the next page. The essence of this version is, that it is free of for us not relevant information like the encoding of the German *f* and has a clear structure.

2.2 CueML ontology

An ontology is needed for automatic extraction and further processing of information. In computer science an ontology is a vocabulary with defined meaning. A description for the use of ontologies can be found in (Berners-Lee et al. 2000).

¹<http://www.tei-c.org/index.xml>


```

<div n="3">
  <head>4. Klare braune Rindflei&#x017F;ch&#x017F;uppe.</head> <lb/
  > <p>Die Bereitung die&#x017F;er braunen Kraftbrühe findet man
    in<lb/> <hi rendition="#aq">A.</hi> No. 12. Zu einer Ge&#
    x017F;ell&#x017F;chaft von 12 Per&#x017F;onen nimmt<lb/> man 6
    Pfund Rindflei&#x017F;ch und 1 Pfund rohen Schinken. Es<lb/>
    werden braune Klöße No. 3 und Schwammklöße darin gemacht.<lb/>
    Auch kann man nach Belieben braunen Sago darin kochen.
  </p>
</div>

```

(a) Example recipe version from (Deutsches Textarchiv 2016)

```

<cue:recipe type="Suppen." rcp-id="B-4">
  <head>Klare braune Rindfleischsuppe.</head>

  <p>Die Bereitung dieser braunen Kraftbrühe findet man in A. No.
    12. Zu einer Gesellschaft von 12 Personen nimmt man 6 Pfund
    Rindfleisch und 1 Pfund rohen Schinken. Es werden braune Klöße
    No. 3 und Schwammklöße darin gemacht. Auch kann man nach
    Belieben braunen Sago darin kochen.
  </p>
</cue:recipe>

```

(b) Example transformed

Figure 2.1: A recipe from our cookbook

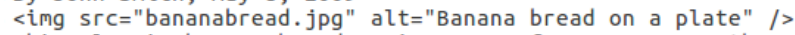
Schema.org/Recipe is an existing ontology for recipes (Schema.org 2016). For example <http://cooking.nytimes.com> and <http://allrecipes.com> use it. Its general usage is shown in fig. 2.2 on the next page and its main purpose is to support search engines as described in (Food Blogger Pro 2014) and (Schema.org 2016) ².

But it is not precise enough for further automatic culinary analysis like extracting the ingredients. As you can see in fig. 2.2 each line from the list of ingredients is marked as an ingredient. This is too inaccurate, because the concrete ingredient is not marked and neither its quantity nor unit. Therefore a computer cannot understand it.

That is why we came up with **culinary editions markup language (cueML)**. It is pronounced like Kümmel, which is the German word for caraway. It is an extension of Schema.org/Recipe. To extend it is a good idea, because this way it keeps all existing advantages of Schema.org/Recipe.

First of all we add the three attributes *quantity*, *unit* and *name*. The name specifies a basis form of the ingredient. That enables a check against an external source, which for example could provide nutrition information or a category like vegetable. The markup for the recipe from fig. 2.2 stays the same, except that the `recipeIngredient` elements get these three extra attributes as shown in listing 2.1.

²at <http://schema.org/docs/datamodel.html>

Mom's World Famous Banana Bread
 By John Smith, May 8, 2009
 />
 This classic banana bread recipe comes from my mom -- the walnuts add a nice texture and flavor to the banana bread.
 Prep Time: 15 minutes
 Cook time: 1 hour
 Yield: 1 loaf
 Tags: Low fat
 Nutrition facts:
 240 calories, 9 grams fat
 Ingredients:
 - 3 or 4 ripe bananas, smashed
 - 1 egg
 - 3/4 cup of sugar
 ...
 Instructions:
 Preheat the oven to 350 degrees. Mix in the ingredients in a bowl. Add the flour last. Pour the mixture into a loaf pan and bake for one hour.
 140 comments:
 From Janel, May 5 -- thank you, great recipe!
 ...

(a) A recipe without markup

```
<div itemscope itemtype="http://schema.org/Recipe">
  <span itemprop="name">Mom's World Famous Banana Bread</span>
  By <span itemprop="author">John Smith</span>,
  <meta itemprop="datePublished" content="2009-05-08">May 8, 2009
  
  <span itemprop="description">This classic banana bread recipe comes
  from my mom -- the walnuts add a nice texture and flavor to the banana
  bread.</span>
  Prep Time: <meta itemprop="prepTime" content="PT15M">15 minutes
  Cook time: <meta itemprop="cookTime" content="PT1H">1 hour
  Yield: <span itemprop="recipeYield">1 loaf</span>
  Tags: <link itemprop="suitableForDiet" href="http://schema.org/LowFatDiet" />Low fat
  <div itemprop="nutrition"
    itemscope itemtype="http://schema.org/NutritionInformation">
    Nutrition facts:
    <span itemprop="calories">240 calories</span>,
    <span itemprop="fatContent">9 grams fat</span>
  </div>
  Ingredients:
  - <span itemprop="recipeIngredient">3 or 4 ripe bananas, smashed</span>
  - <span itemprop="recipeIngredient">1 egg</span>
  - <span itemprop="recipeIngredient">3/4 cup of sugar</span>
  ...
  Instructions:
  <span itemprop="recipeInstructions">
  Preheat the oven to 350 degrees. Mix in the ingredients in a bowl. Add
  the flour last. Pour the mixture into a loaf pan and bake for one hour.
  </span>
  140 comments:
  <div itemprop="interactionStatistic" itemscope itemtype="http://schema.org/InteractionCounter">
    <meta itemprop="interactionType" content="http://schema.org/CommentAction" />
    <meta itemprop="userInteractionCount" content="140" />
  </div>
  From Janel, May 5 -- thank you, great recipe!
  ...
</div>
```

(b) The same recipe enriched with markup

Figure 2.2: Schema.org/Recipe example from (Schema.org 2016)

```

Ingredients:
- <span itemp="recipeIngredient" quantity="3-4" name="banana">3
  or 4 ripe bananas, smashed</span>
- <span itemp="recipeIngredient" quantity="1" name="egg">1 egg</
  span>
- <span itemp="recipeIngredient" quantity="0.75" unit="cup" name
  ="sugar">3/4 cup of sugar</span>

```

Listing 2.1: Example for cueML

Additionally we add an attribute *isOptional*, because we think it is not appropriate to simply state sugar is an ingredient in the case of "sugar if not sweet enough".

Sometimes an ingredient is composed. For example "potato dumplings" is a reference to a complete different recipe. For capturing this, we add an attribute *reference*, which points to another source. We also allow an element *link* with an attribute *target*. This is just a look up reference for example to general cooking advice.

Last we also noticed, that ingredients can be alternatives to each other like in "hollandaise sauce or butter, mustard and a squeeze of lemon juice". Therefore we add an element *recipeIngredientAlternations* having arbitrary child elements *alt*, which enclose *recipeIngredient* elements.

An example is provided in appendix B. The complete cueML ontology is defined through a relax ng grammar, which can be found in appendix C. Due to the point, that Schema.org does not provide a grammar for Schema.org/Recipe, cueML includes also the rules for it. Personally I was surprised that they do not provide an official grammar, because that would provide a clear definition as well as an easy validation method. That is why we provide a grammar. But they state themselves, that "some data is better than none", meaning, that they want to tolerate wrong meta data for reducing the risk of getting no meta data at all. They state furthermore, that it makes it more easy to extend the language (Schema.org 2016)³. Obvious you cannot break, what is not defined.

2.3 Applying cueML to our recipes

Due to the point, that we wanted to extend Schema.org/Recipe, cueML should be applied to the list of ingredients as the former does. But our recipes have only a direction text, as you can see in fig. 2.1. CueML as well as Schema.org/Recipe is not well suited for direction text. The ingredient phrases are not always tied closely together. Besides they repeat themselves in the text and sometimes they appear in the form of "like the previous one but don't use onions". These points make it hard to apply cueML directly to the direction text. Therefore we decided to collect them in an extra *meta*-element. Having extra sections for recipe yield, cook time and list of ingredients is way more nicer to read than only one direction text anyway. Appendix B shows a full example usage of cueML. A test with the testing tool for structured data from google⁴ reveals, that all extracting, which is already in place for Schema.org/Recipe, still works.

³at <http://schema.org/docs/datamodel.html>

⁴<https://search.google.com/structured-data/testing-tool>

2.4 Need for automation

As mentioned in (Erdmann et al. 2000), manual tagging is time consuming and error prone. The approach to extract ingredients from recipes, which we will present in section 3.3, emphasises that. They state, that they need 50-100 fields per recipe. And in the evaluation of their approach they discovered mistakes done in manual tagging.

For tagging one recipe in our cookbook I need about 5 minutes. That means, that I would need more than 2 weeks, for tagging only the recipes from our cookbook. Building a big data pool this way is very time consuming and therefore expensive.

Hence automation, which has to be configured only once and can be applied to many resources afterwards, is clearly preferable.

Chapter 3

Related Work

In general there exists many works about extracting useful information from textual and unstructured resources. The superordinate term for this field of research is Text Mining. It was first mentioned in (Feldman and Dagan 1995) and an overview can be found in (Hotho et al. 2005).

The algorithms for extracting useful information depend highly on existing semi-structures, which can be taken advantage of. Here we present existing algorithms, which we found in the domain of cooking, and distinct their effort from this thesis. But before that we define precision and recall.

3.1 Precision & Recall

Precision and recall are metrics, which measure the quality of an information extracting algorithm.

$$Precision = \frac{\#(retrieved \cap relevant)}{\#retrieved}, \quad Recall = \frac{\#(retrieved \cap relevant)}{\#relevant} \quad (3.1)$$

They are defined as shown in eq. (3.1) according to (Hotho et al. 2005). A high precision states, that the algorithm does only find relevant information as intended. The ideal precision of one would mean, that all extracted data were useful. A high recall states, that the algorithm finds many of the total relevant information. The perfect score of one would mean, that all relevant information were found. Both are needed for the evaluation of an algorithm. If only considering precision, the algorithm could only find the information, which are obvious relevant and therefore find only view information, but having a high score this way. On the other side, if only considering recall, the algorithm could return everything. This way its score would have the perfect value of one. But both algorithm are obvious not good, which gets covered by a low value of the other formula.

3.2 Skip The Pizza

(*Skip The Pizza* 2012) is a project described on WordPress.org. The author wants to combine his two hobbies cooking and software engineering. For being able to answer questions like "How many ingredients does a typical recipe consist of?" or

"Which are the most frequent ingredients?", he extracts the ingredients of recipes from an open source platform at http://recipes.wikia.com/wiki/Recipes_Wiki.

The recipes have a consistent internal representation, which is shown in listing 3.1. The semi-structure, that after `== Ingredients ==` comes a list of ingredients, can be recognized easily. Per line is one ingredient enclosed within `[[ingredient name]]`. Using this semi-structure a regular expression is already good enough for extracting the ingredients from these recipes.

```
* Makes 6 to 8 servings

== Ingredients ==
* 2 tbsp extra virgin [[olive oil]]
* 3 cloves [[garlic]], finely chopped
[...]

== Directions ==
Heat olive oil and garlic in large skillet over low heat until
garlic begins to sizzle.
Add tomatoes, [...]

[[Category:Cathy's Recipes]]
[[Category:Garlic Recipes]]
[...]
```

Listing 3.1: Shortened example recipe from
http://recipes.wikia.com/wiki/Recipes_Wiki

3.3 Extracting Structured Data From Recipes Using Conditional Random Fields

The New York Times provides a cooking website with recipes¹. Their recipes are enriched with Schema.org/Recipes. For providing a recipe recommendation system based on ingredients, you have to extract the exact ingredients from a recipe, which is not enabled through this schema, as already discussed in section 2.2. Nevertheless they are able to extract them automatically. They use the provided structure from Schema.org/Recipe, that each ingredient phrase from the list of ingredients is marked as `recipeIngredient`, and Conditional Random Fields (CRF) for that. Their approach is described in (The New York Times 2015). Hence we introduce here CRF first and afterwards outline their implementation.

3.3.1 Conditional Random Fields

Given a vector of words, CRF wants to predict a suitable vector of labels. For example when the vector of words is *[1 tablespoon salt]*, we want to predict *[QUANTITY, UNIT, INGREDIENT]*, meaning 1 is a quantity, tablespoon a unit and salt an ingredient.

A detailed introduction to CRF can be found in (Sutton and McCallum 2012). Here we only want to give a quick overview about linear-chain CRF, because that is

¹<http://cooking.nytimes.com/>

the algorithm the New York Times uses. Therefore, when we write CRF, we mean linear-chain CRF.

Its starting point are vectors of words X , which have already correct vectors of labels Y . Such related sets of vectors are called training data. A joint probability distribution can be extracted from this training data, which states how likely a vector of words has a corresponding vector of labels. Taking the simplified assumption, that each label depends only on the previous label and that the current word depends only on the current label, leads to eq. (3.2). This can always be transformed into the form of eq. (3.3). The division with $Z(X, Y)$ ensures, that the value of $p(X, Y)$ is between zero and one. $1_{condition}$ is a function, which is one if the condition is true and zero otherwise. Smart indexing leads to eq. (3.4). Each f_k is called a feature function. The calculation of the Θ_k 's is a mathematical optimisation problem. Note that there is very likely no exact solution due to the simplified assumption of eq. (3.2).

$$p(X, Y) = \prod_{t=1}^T p(y_t | y_{t-1}) * p(x_t | y_t), \quad T = \#X \quad (3.2)$$

$$\begin{aligned} p(X, Y) &= \frac{1}{Z(X, Y)} \prod_{t=1}^T \exp\left(\sum_{i,j \in S} \Theta_{i,j} * 1_{y_t=i} * 1_{y_{t-1}=j} + \sum_{i \in S} \sum_{j \in O} \mu_{o,i} * 1_{y=i} * 1_{x_t=o}\right), \\ Z(X, Y) &= \sum_X \sum_Y \prod_{t=1}^T \exp\left(\sum_{i,j \in S} \Theta_{i,j} * 1_{y_t=i} * 1_{y_{t-1}=j} + \sum_{i \in S} \sum_{j \in O} \mu_{o,i} * 1_{y=i} * 1_{x_t=o}\right), \\ &\quad S = \text{all possible labels}, \quad O = \text{all possible words} \end{aligned} \quad (3.3)$$

$$\begin{aligned} p(X, Y) &= \frac{1}{Z(X, Y)} \prod_{t=1}^T \exp\left(\sum_{k=1}^K \Theta_k * f_k(y_t, y_{t-1}, x_t)\right), \\ Z(X, Y) &= \sum_Y \prod_{t=1}^T \exp\left(\sum_{k=1}^K \Theta_k * f_k(y_t, y_{t-1}, x_t)\right) \end{aligned} \quad (3.4)$$

A joint probability distribution can always be transformed into a conditional probability as shown in eq. (3.5).

$$p(Y|X) = \frac{p(X, Y)}{\sum_{Y' \in S} p(Y', X)} \quad (3.5)$$

The described model so far is a Hidden Markov Model. In a CRF you are also allowed to take only a subset of these features, what can improve performance without losing accuracy. Additional improvement in a CRF can very likely be achieved by further custom feature functions, which may depend on the whole vector of words, instead of only the current word identity. For example some custom feature functions could be:

- $f_{k+1}(y_t, y_{t-1}, X) = 1_{x_t \text{ startswith upper case}}$
- $f_{k+2}(y_t, y_{t-1}, X) = 1_{x_t \text{ is in a domain specific dictionary}}$
- $f_{k+3}(y_t, y_{t-1}, X) = 1_{x_t \text{ is in a domain specific dictionary and } x_{t-1} \text{ is an article}}$

Putting all together leads to definition 3.1. Note that after plugging eq. (3.4) into eq. (3.5) Z now only depends on X .

Definition 3.1 (linear-chain Conditional Random Field) A linear-chain Conditional Random Field is a conditional distribution of the form:

$$p(Y|X) = \frac{1}{Z(X)} \prod_{t=1}^T \exp(\sum_{k=1}^K \Theta_k * f_k(y_t, y_{t-1}, X)), \text{ where}$$

$$Z(X) = \sum_S \prod_{t=1}^T \exp(\sum_{k=1}^K \Theta_k * f_k(y_t, y_{t-1}, X)) \text{ and}$$

Y vector of labels, X vector of words, S set of all possible labels

Having a CRF in place, a natural prediction function is shown in eq. (3.6). The calculation of $prediction(X)$ can be done in $(\#S)^2 * \#X$ through dynamic programming.

$$prediction(X) = argmax_Y (P(Y|X)) \quad (3.6)$$

3.3.2 Implementation of New York Times

As starting point the New York Times have manually specified labels for over 130,000 ingredient phrases, which they use as training data. They are extracted out of their website, where their recipes are already enriched with Schema.org/Recipe.

One example ingredient phrase is shown in listing 3.2. They use CRF++² as library, which is an implementation of CRF. The first word per column is the actual word. The last word is the label, which should be predicted in IOB2 format. It states, that the beginning of an entity gets prefixed with *B-* and the continuation with *I-*, as you can see in the last three words. Words, which do not belong to a relevant entity, get labelled with *OTHER*. The labels in between are custom features.

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

Listing 3.2: Extract of the training data for New York Times CRF

The feature functions, which the CRF should use, are not specified through functions in a concrete programming language. Instead they get described through templates of the format `U00:%x[row,column]`, as shown in listing 3.3. The row specifies the targeted word in relative position to itself and the column specifies the absolute position. Therefore when the actual word is mushrooms the data from listing 3.2 would be expended to the values in listing 3.4.

²<https://taku910.github.io/crfpp/>


```

# 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

```

Listing 3.3: Feature templates for New York Times CRF

```

pound
shiitake
mushrooms
,
stemmed
I4
L12
NoCAP

NoPAREN
NoPAREN
NoPAREN
NoPAREN
NoPAREN

mushrooms/L12
I4 /L12
mushrooms/NoCAP
mushrooms/NoPAREN
mushrooms/I4

# Bigram
I-NAME B-Name

```

Listing 3.4: Derived value when the current word in listing 3.2 is mushrooms

Feature functions are extracted from these templates as shown in listing 3.5. The B-template takes the transition from y_{t-1} to y_t into account. Note that for example the custom feature function, if a word is inside parentheses, gets implemented through adding a label YesPAREN/NoPAREN to the trainings data in the fourth column in a preprocessing step and specifying the feature template U10%[0,4]. Also note, that for each such template $\#(all\ possible\ labels) * \#(all\ possible\ words)$ different feature functions gets created. For the B-template $\#(all\ possible\ labels) * \#(all\ possible\ labels) * \#(all\ possible\ words)$ different feature functions gets created.

U02:%[0,0]	$\rightarrow 1_{y_t=I-Name\ and\ x_t=mushrooms}$
U02:%[-2,4]	$\rightarrow 1_{y_t=I-Name\ and\ x_{t-2}\ 4th\ label\ is\ NoPAREN}$
U13:%x[0,0]/%x[0,2]	$\rightarrow 1_{y_t=I-Name\ and\ x_t=mushrooms\ and\ x_t\ 2nd\ label\ is\ L12}$
B	$\rightarrow 1_{y_t=A\ and\ y_{t-1}=B\ for\ each\ A, B \in possible\ prediction\ labels}$

Listing 3.5: Extracting of feature functions from templates

Hence the only features the New York Times take into account for prediction are the word identity (column 0), the index of the current word (column 1), the length of the sentence (column 2, rounded to 4, 8, 12, 16 or 20), if the word starts capitalized (column 3) and if the word is inside parenthesis (column 4).

When applying their algorithm to 481 example recipes, they get 89% sentence-

level accuracy, meaning that they get 9 out of 10 ingredient phrases completely right. They tested only so few recipes, although they had over 130,000 ready labelled recipes, because they found no way for automatic evaluation. Their problem was, that there were too many ambiguous phrases and too many mistakes in the manually labelled data.

3.4 Domain Specific Information Extraction for Semantic Annotation

(Ahmed 2009) is a diploma thesis about extracting ingredients and their further processing from recipes. Their algorithm could be divided into two main parts.

First to check every word, if it is in a dictionary of ingredients respectively a dictionary of actions and label it accordingly. For keeping the dictionaries as small as possible they do a morphological Analysis and only store the lemmas of the words. The second main part, and more sophisticated task, is to identify, which action should be applied to which ingredient. They have a rule based approach and a dependency parsing based approach for that.

The rule based approach is to do a part of speech tagging (POS) first and afterwards try to apply a small set of rules, which define a context free grammar, with predefined meaning. Listing 3.6 shows an example, meaning buttermilk and bananas should be added.

```
# Example sentence
add buttermilk and bananas

# Example sentence with POS and dictionary-label
add[ACT] buttermilk[ING] and[CC] bananas[ING]

# Apply rule VP -> ACT NP (,NP)* (CC NP)?
# Predefined meaning of the rule: Apply ACT to all ING
-> ACT NP CC NP
# Apply rule NP -> DT? JJ* ING twice
-> ACT ING CC ING

# With:
# VP, NP ∈ non-terminal symbols, CC ∈ Conjunctions, DT ∈ Determiner,
# ACT := action, ING := ingredient
```

Listing 3.6: Rule based example

The second approach is to do a dependency based parsing, which represents the semantic structure of a sentence in a tree like format. For example fig. 3.1 represents, that the subject of like is John and the liked object is Apple. The format as well as building the tree is way more complex than the previous simple rules, but the tree can be build by already existing tools like the Stanford Parser³. The semantic structure of the sentences give strong evidence, which action should be applied to which ingredient.

³<http://nlp.stanford.edu/software/lex-parser.shtml>

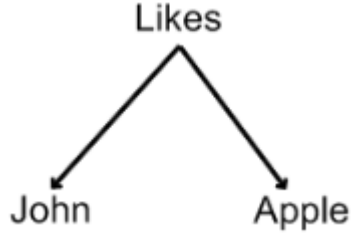


Figure 3.1: John likes apple (Ahmed 2009)

In their evaluation they apply these two variants to 43 randomly selected recipes from the internet. Their precision and recall are presented in table 3.1. Even the more sophisticated dependency based approach has only a recall of 64%. This is due to a lack of mapping actions to their corresponding ingredients.

	Precision	Recall
Rule based	97.39%	51.54%
Dependency Based	95.4%	64.12%

Table 3.1: Evaluation Domain Specific Information Extraction for Semantic Annotation

3.5 Distinction to this work

(*Skip The Pizza* 2012) exploits a strict semi-structure, which is not given in our very unstructured cookbook. Therefore we need more advanced algorithm than regular expressions.

(The New York Times 2015) take advantage of the point, that they have already marked the ingredient phrases from the list of ingredients. But our cookbook has no list of ingredients. Nevertheless, the ingredient phrases could be extracted from plain text through a CRF algorithm first and afterwards we could use the algorithm of (The New York Times 2015). However we have to consider, that we do not have near as 130,000 training data. Our lack of training data could be caught up by more sophisticated custom feature functions. Maybe good feature functions even enable to extract all wanted data in one CRF algorithm.

(Ahmed 2009) only uses plain text. But they have a bad recall, due to the point, that they are not good in mapping actions to their corresponding ingredients. In contrast to mapping actions to ingredients, we want to map quantities to units and these in turn to ingredients, which is more complex and could therefore lead to an even worse recall. Though the dictionary check, enabled by a morphological analysis, from their algorithm first main part, could be very good feature functions.

Chapter 4

Development of recipe parser

4.1 Starting position

-plain text -altes deutsch / alte Zutaten -how to test

4.2 Overall picture

Git-tag

4.2.1 Workflow

4.2.2 Features

4.2.3 Evaluation

-Precision -Recall

4.3 First Iteration: Basis CRF

Git-tag

4.3.1 Workflow

4.3.2 Features

4.3.3 Evaluation

4.4 Final recipe parser

4.4.1 Workflow

4.4.2 Features

4.4.3 Evaluation

Chapter 5

Discussion

- Übertragung auf beliebige Bücher (Wenn Buch/Webseite hat bereits Zutatenliste erste Phase entfällt)

- TDD / früh Plausibilitäts-Überprüfungen - Tagen als Hi-Wi war ne dumme Aktion‘ (insbesondere fürs Schemata, Mengenangaben und Namen historische Recherche nötig (B-17 bzw appendix B cook time not clear, Wurzelwerk, braunes Gewürz, Engl. Soja))

5.1 Power of machine readable data

Machine readable data are very powerful. But *with great power comes great responsibility*¹. In the context of a recipe parser this might be a little bit exaggerated. But specially in mind of the global surveillance disclosures denounced by Edward Snowden with still uncertain dimension, I think it is important to have a consciousness for what can be done. Therefore I want to think about, what can be done through innocent looking machine readable tags.

For the good there exists already much effort for services, which require being able to extract ingredient from recipes.

(e.g. Teng et al. 2012 or Ueda et al. 2011).

Further more, having a huge machine-readable base of recipes and its ingredients, can also provide insights in sociological research. For example in (Ahn et al. 2011) is a comparison between American and Asian kitchen based on about 56.000 recipes.

There are many more interesting questions, which could be analysed like a historic analysis of the development and changes of cooking. Occurrences of non-local ingredients or meals are evidence for inter cultural exchange and globalisation. More expensive ingredients could be an indication for prosperity, while very simple kitchen for poverty or even wartimes.

In the bad (The Washington Post 2016) "schwacher vegetarier"

¹A well known proverb which probably has its origin from the French National Convention during the period of French Revolution. (*With Great Power Comes Great Responsibility* 2015)

Chapter 6

Summary

Appendix A

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

Appendix B

Full example use of cueML

```
<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 Ochsenkaumen und eine geräucherte 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 braunem Gewürz, ein
    Paar Messerspitzen Cayenne-Pfeffer, einige Kalbsmidder in Stü
    ckchen 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 1/4 Stunde gekocht hat, kommen
    noch Klöße von Kalbfleisch, einige hart gekochte Eier in Wü
    rfel geschnitten, ein Paar Eßlöffel Engl. Soja hinzu, und wenn
    die Klößchen einige Minuten gekocht haben, 1/2 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 wegbleiben, 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>
```

Listing B.1: Without cueML

```
<div itemscope itemtype="http://schema.org/Recipe"http://cueML.org"
  >
  <recipe type="Suppen." rcp-id="B-16">
    <head><span itemprop="name">Mock Turtle Suppe</span></head>

    <meta>
      <span itemprop="recipeYield" quantity="24-30" unit="Personen"
        >24-30 Personen</span>

      <span itemprop="recipeIngredient" name="Bouillon" reference="
        #Bouillon">Bouillon</span>
      <span itemprop="recipeIngredient" name="Rindfleisch" quantity
        ="8-10" unit="Pfund">8-10 Pfund Rindfleisch</span>
```



```

<span itemprop="recipeIngredient">Wurzelwerk</span>
<span itemprop="recipeIngredient" name="Kalbskopf" quantity="1">1 Kalbskopf</span>
<span itemprop="recipeIngredient" name="Schweineschnauze" quantity="1">1 Schweineschnauze</span>
<span itemprop="recipeIngredient" name="Schweineohr">Schweineohren</span>
<span itemprop="recipeIngredient" name="Ochsengaumen" quantity="1">1 Ochsengaumen</span>
<span itemprop="recipeIngredient" name="Ochsenzunge" quantity="1">1 geräucherte Ochsenzunge</span>
<span itemprop="recipeIngredient">braunes Gewürz</span>
<span itemprop="recipeIngredient" name="Cayennepfeffer" quantity="vague" unit="Messersptize">ein Paar Messerspitzen Cayenne-Pfeffer</span>
<span itemprop="recipeIngredient" name="Kalbsmidder" quantity="vague" reference="#A-16">einige Kalbsmidder</span>
<span itemprop="recipeIngredient" name="Saucisson">kleine Saucissen</span>
<span itemprop="recipeIngredient" name="Butter">Butter</span>
<span itemprop="recipeIngredient" name="Mehl">Mehl</span>
<span itemprop="recipeIngredient" name="Kalbfleischklöße" reference="#L-4">Klöße vom Kalbfleisch</span>
<span itemprop="recipeIngredient" name="Ei" quantity="vague">einige hart gekochte Eier</span>

<span itemprop="recipeIngredient" name="Auster" isOptional="True">Austern, wenn man sie haben kann</span>
<span itemprop="recipeIngredient" quantity="vague" unit="EL" isOptional="True">ein Paar Eßlöffel Engl. Soja</span>

<recipeIngredientAlternations>
  <alt>
    <span itemprop="recipeIngredient" name="Madeira" quantity="0.5" unit="Flasche">Madeira</span>
  </alt>
  <alt>
    <span itemprop="recipeIngredient" name="weißen_Franzwein">weißen Franzwein</span>
    <span itemprop="recipeIngredient" name="Rum">etwas Rum</span>
  </alt>
</recipeIngredientAlternations>
</meta>

<span itemprop="recipeInstructions">
  <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 Ochsengaumen und eine geräucherte Ochsenzunge zu Feuer und kocht dies Alles gahr, aber nicht zu weich. Kalt, schneidet man es in kleine, 1 änglich viereckige Stückchen, gibt das Fleisch in die Bouillon, nebst braunem Gewürz, ein Paar Messerspitzen Cayenne-Pfeffer, einige Kalbsmidder 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 1/4 Stunde gekocht hat, kommen noch
Klöße von Kalbfleisch, einige hart gekochte Eier in Würfel
geschnitten, ein Paar Eßlöffel Engl. Soja hinzu, und wenn
die Klößchen einige Minuten gekocht haben, 1/2 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 wegbleiben, 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>
</span>
<recipe>
</div>
```

Listing B.2: Enriched with cueML

Appendix C

RELAX NG grammar for cueML

Bibliography

- Ahmed, Zeeshan (2009). “Domain Specific Information Extraction for Semantic Annotation”. Diploma thesis. Charles University in Prague, Czech Republic and University of Nancy 2 in Nancy, France.
- Ahn, Yong-Yeol et al. (2011). “Flavor network and the principles of food pairing”. In: *Scientific Reports*. URL: <http://www.nature.com/articles/srep00196>.
- Berners-Lee, T., J. Hendler, and O. Lassila (2000). “Semantic web”. In: *Scientific American*, 1(1):68–88.
- Deutsches Textarchiv, Berlin-Brandenburgische Akademie der Wissenschaften (2016). *Davidis, Henriette: Praktisches Kochbuch für die gewöhnliche und feinere Küche. 4. Aufl. Bielefeld, 1849*. URL: http://www.deutschestextarchiv.de/book/show/davidis_kochbuch_1849 (visited on 10/30/2016).
- Erdmann, M. et al. (2000). “From Manual to Semi-automatic Semantic Annotation: About Ontology-based Text Annotation Tools”. In: *Scientific American*, 1(1):68–88.
- Feldman, Ronen and Ido Dagan (1995). “Knowledge Discovery in Textual Databases (KDT)”. In: *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*. KDD’95. Montréal, Québec, Canada: AAAI Press, pp. 112–117. URL: <http://dl.acm.org/citation.cfm?id=3001335.3001354>.
- Food Blogger Pro (2014). *What is recipe schema and how does it impact my food blog?* URL: <https://www.foodbloggerpro.com/blog/article/what-is-recipe-schema/> (visited on 11/06/2016).
- Hotho, Andreas, Andreas Nürnberger, and Gerhard Paaß (2005). “A brief survey of text mining”. In: *LDV Forum - GLDV Journal for Computational Linguistics and Language Technology*.
- Schema.org (2016). *Schema.org*. URL: <https://schema.org/> (visited on 10/30/2016).
- Skip The Pizza (2012). URL: <http://skipthepizza.com/blog/analyzing-the-ingredients-of-29200-recipes> (visited on 10/28/2016).
- Sutton, Charles and Andrew McCallum (2012). “An Introduction to Conditional Random Fields”. In: *Found. Trends Mach. Learn.* 4.4, pp. 267–373. ISSN: 1935-8237. DOI: 10.1561/22000000013. URL: <http://dx.doi.org/10.1561/22000000013>.
- Teng, Chun-Yuen, Yu-Ru Lin, and Lada A. Adamic (2012). “Recipe Recommendation Using Ingredient Networks”. In: *Proceedings of the 4th Annual ACM Web Science Conference*. WebSci ’12. New York, NY, USA: ACM, pp. 298–307. ISBN: 978-1-4503-1228-8. DOI: 10.1145/2380718.2380757. URL: <http://doi.acm.org/10.1145/2380718.2380757>.
- The New York Times (2015). *Extracting Structured Data From Recipes Using Conditional Random Fields*. URL: <http://open.blogs.nytimes.com/2015/04/09/>

extracting-structured-data-from-recipes-using-conditional-random-fields/?_r=1 (visited on 10/30/2016).

The Washington Post (2016). *Hillary Clinton's health just became a real issue in the presidential campaign*. URL: <https://www.washingtonpost.com/news/the-fix/wp/2016/09/11/hillary-clintons-health-just-became-a-real-issue-in-the-presidential-campaign/> (visited on 11/02/2016).

Ueda, Mayumi, Mari Takahata, and Shinsuke Nakajima (2011). "User's Food Preference Extraction for Personalized Cooking Recipe Recommendation". In: *Proceedings of the Second International Conference on Semantic Personalized Information Management: Retrieval and Recommendation - Volume 781*. SPIM'11. Aachen, Germany, Germany: CEUR-WS.org, pp. 98–105. URL: <http://dl.acm.org/citation.cfm?id=2887675.2887686>.

With Great Power Comes Great Responsibility (2015). URL: <http://quoteinvestigator.com/2015/07/23/great-power/> (visited on 11/02/2016).