DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2016

Information extraction from text recipes in a web format

JOHAN STORBY

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF COMPUTER SCIENCE AND COMMUNICATION

KTH Computer Science
and Communication

Information extraction from text recipes in a web format

JOHAN STORBY

Master's Thesis at CSC
Supervisor: Olov Engwall
Examiner: Viggo Kann

22 June 2016

Abstract

Searching the Internet for recipes to find interesting ideas for meals to prepare is getting increasingly popular. It can however be difficult to find a recipe for a dish that can be prepared with the items someone has available at home. In this thesis a solution to a part of that problem will be presented.

This thesis will investigate a method for extracting the various parts of a recipe from the Internet in order to save them and build a searchable database of recipes where users can search for recipes based on the ingredients they have available. The system works for both English and Swedish and is able identify both languages.

This is a problem within Natural Language Processing and the subfield Information Extraction. To solve the Information Extraction problem rule-based techniques based on Named Entity Recognition, Content Extraction and general rule-based extraction are used.

The results indicate a generally good but not flawless functionality. For English the rule-based algorithm achieved an F1-score of 83.8% for ingredient identification, 94.5% for identification of cooking instructions and an accuracy of 88.0% and 96.4% for cooking time and number of portions respectively. For Swedish the ingredient identification worked slightly better but the other parts worked slightly worse.

The results are comparable to the results of other similar methods and can hence be considered good, they are however not good enough for the system to be used independently without a supervising human.

Referat

Informationsextraktion ur textrecept i webbformat

Att söka på Internet efter recept för att hitta intressanta idéer till måltider att laga blir allt populärare. Det kan dock vara svårt att hitta ett recept till en maträtt som kan tillagas med råvarorna som finns hemma. I detta examensarbete kommer en lösning på en del av detta problem att presenteras.

Detta examensarbete undersöker en metod för att extrahera de olika delarna av ett recept från Internet för att spara dem och fylla en sökbar databas av recept där användarna kan söka efter recept baserat på de ingredienser som de har till förfogande. Systemet fungerar för både engelska och svenska och kan identifiera båda språken.

Detta är ett problem inom språkteknologi och delfältet informationsextraktion. För att lösa informationsextraktionsproblemet använder vi regelbaserade metoder baserade på entitetsigenkänning, metoder för extraktion av brödtext samt allmäna regelbaserade extraktionsmetoder.

Resultaten visar på en generellt bra men inte felfri funktionalitet. För engelska har den regelbaserade algoritmen uppnått ett F1-värde av 83,8 % för ingrediensidentifiering, 94,5 % för identifiering av tillagningsinstruktioner och en träffsäkerhet på 88,0 % och 96,4 % för tillagningstid och antal portioner. För svenska fungerade ingrediensidentifieringen något bättre än för engelska men de andra delarna fungerade något sämre.

Resultaten är jämförbara med resultaten för andra liknande metoder och kan därmed betraktas som goda, de är dock inte tillräckligt bra för att systemet skall kunna användas självständigt utan en övervakande människa.

Contents

1 Introduction **1**
1.1 Goal . . . 1
1.2 Objective . . . 1
1.3 Scientific question . . . 2
1.4 Ethical, social and environmental issues . . . 2
1.5 Limitations . . . 2

2 Background **3**
2.1 Natural Language Processing . . . 3
2.1.1 Information Extraction . . . 3
Rule based methods . . . 4
Machine learning methods . . . 4
2.1.2 Evaluation measures . . . 5
2.1.3 Web environment . . . 6
DOM tree . . . 6
2.1.4 Stemming . . . 7
2.1.5 Lemmatization . . . 8
2.1.6 Tokenisation . . . 8
2.1.7 n-grams . . . 9
2.2 Language Identification . . . 9
2.3 Part-of-speech tagging . . . 10
2.4 Named Entity Recognition . . . 11
2.5 Content Extraction . . . 13
2.6 Conclusions of background study . . . 16

3 Method **17**
3.1 Language Identification . . . 17
3.1.1 Generating the n-gram array . . . 17
3.1.2 Cosine similarity . . . 18
3.1.3 Corpus . . . 19
3.2 Information Extraction . . . 19
3.2.1 Existing system . . . 19
3.2.2 Gazetteers . . . 19

3.2.3 Preprocessing 20
Tokenisation 20
Reverse stemming 20
3.2.4 Identifying ingredients 21
Strict ingredients 22
3.2.5 Identifying cooking instructions 22
3.2.6 Identifying cooking time 23
3.2.7 Identifying number of portions 24
3.3 Experiment 25
3.3.1 Recipes 25

4 Results **27**
4.1 Language Identification 27
4.2 Information Extraction 27

5 Discussion **31**
5.1 Experiment 31
5.2 Language Identification 32
5.3 Information Extraction 32
5.3.1 Ingredient Identification 32
5.3.2 Extraction of Cooking Instructions 33
5.3.3 Cooking time and number of portions 35

6 Conclusions **37**
6.1 Future work 38

References **39**

Appendices **40**

A Recipes **41**
A.1 English 41
A.2 Swedish 42

Chapter 1

Introduction

This report will show the result of an investigation of a method for extracting the different parts of a recipe in text form from a web page. The report will cover various methods for extraction of the different parts using different methods within the field of *Natural Language Processing*, and specifically *Information Extraction* from web documents.

This report should be able to contribute new methods and information regarding extraction and how information extraction problems can be solved for a specific type of text in a particular format, in this case an HTML format.

1.1 Goal

The goal is to investigate different ways of extracting the relevant information from a recipe in an HTML format on the web. The information to be extracted is title, ingredients along with measurements and other related information such as unit, cooking instructions, cooking time and number of portions. The system should work for both English and Swedish and be able to detect both languages.

The purpose of extracting those parts from a recipe is to fill a database with recipes where it is possible to search for recipes based on ingredients and filter by other factors such as cooking time.

In order for this database to be built correctly it is important that the system is as accurate as possible and that no relevant information is left out.

There are many different ways in which this information can be presented and a system has to be able to identify this information from various different sources using a general method which should be general enough to be applicable to multiple systems, while still not miss any information or include erroneous information.

1.2 Objective

There is a system with some functionality already implemented. For this thesis a brand new system will be developed, and that new system will be tested alongside with the existing system, which will receive some small improvements. The information extraction consists of many smaller problems, which will be evaluated separately, and hence the final solution may be a combination of both solutions.

1.3 Scientific question

Can a rule-based system successfully be used to identify and extract the different parts of a recipe in HTML-format, as well as extracting additional information about ingredients such as amount and unit?

1.4 Ethical, social and environmental issues

There are no obvious major ethical or societal effects to consider in this work. What to consider is that a system like this could gather large amounts of recipes from external sources, and that those external sources might not appreciate using their content to fill a database of recipes. Recipes themselves do not fall under the copyright laws unless they can be considered to have an “artistic quality“ and the recipe then has to be written in an artistic manner rather than the usual descriptive fashion [1, 2]. Images related to the recipes do fall under the copyright laws and are hence not to be extracted.

There are no direct environmental issues to consider in this project, however there are some indirect environmental issues. Successful results in this experiment may be beneficial for the environment as food waste may decrease if people can plan their meals better with the final product that this experiment will be a part of. On the other hand, being able to find new recipes may incline people to buy products that they otherwise would not, and hence food waste could increase.

For the same reasons, this may be both financially positive or negative for the user depending on how the application is used. Neither of these effect are very large and the overall impact is not very large.

1.5 Limitations

This project will only investigate methods for extracting information from recipes, extracting from other types of documents will not be considered. Only recipes in English and Swedish will be used when developing and testing. The only parts to be extracted are cooking instructions, title, amount of portions, cooking time and ingredients together with information attached to the ingredients such as amount, unit and an associated comment where applicable.

Developing a database for storing is not necessary as this is already done, and the ingredients in this database can be used as a *gazetteer* when matching ingredients. Further development of this database is not included in this project.

Chapter 2

Background

This section describes the background and related work to this study. Initially the basics around Natural Language Processing and Information Extraction are covered and then the topics are described in more detail and finally the subject of Content Extraction is covered.

2.1 Natural Language Processing

Natural Language Processing (NLP) is a field within computer science which deals with natural language, either in spoken or written form. Natural languages are the languages that are used by humans to communicate, for example English. Formal languages on the other hand are more strictly defined and are used in cases where strict formality is required, in for example mathematics or computer programming. Natural languages differ from formal languages in that the rules about how things are expressed in the language are not strict and absolute, and hence natural languages are much more difficult for an algorithm to process than formal languages. This is why natural language problems are more difficult than many other computer science problems [3].

Natural Language Processing is one of the oldest fields of computer science. Initially the main applications were in military intelligence, but today the field has shifted to a broader context and mostly of commercial use.

There are many NLP subfields, including machine translation, text summarization, spell checking, information retrieval and speech recognition. This report however will focus on some parts of another subfield; *information extraction*, and in particular *named entity recognition* and *content extraction.*

2.1.1 Information Extraction

Information Extraction is a subfield within Natural Language Processing and is the task of extracting structured information from natural language [4], in this case a cooking recipe on a web page.

Information extraction can work in different ways, and different approaches may work better depending on what type of information is to be extracted and what type of document the information is extracted from. There are two different kinds of methods, rule based approaches and statistical methods, also known as machine learning, and hybrid methods which are a mix of those two [5].

Rule based methods

Rule based methods work by using a set of predefined rules, which in the case of information extraction may look for certain words, a certain sequence of words, or in the case of a web document specific HTML-tags. These rules have to be hand crafted by a human who has knowledge about how the data is structured. The advantage of rule based systems is that there is no need to collect training data and it can be easier to tailor the system to one's needs. If a specific part is not working correctly that part can be corrected. The downside of a rule-based system is that for a more complex problem it can be very difficult to craft rules that will work under general cases and it may be very difficult to craft rules for all possible outlier situations [5].

Machine learning methods

Statistical methods, also known as machine learning, work by using mathematical methods to recognize patterns and learning how to recognize different types of data. There are two different kinds of machine learning, supervised learning and unsupervised learning. Supervised learning requires labeled data, meaning a set of data where the data comes together with a label, or a "correct answer" which is used as training data to later classify other pieces of data according to the training data. Unsupervised learning does not require any labeled training data, hence it can not be used to classify information, but rather be used for clustering, finding similar data, and approaches related to that. There is also the possibility of a mix between those two, known as semi-supervised learning or a hybrid method, which uses mostly non-labeled data, but also requires a small amount of labeled data [5].

The advantages of a machine learning method is that it can learn more complex patterns than rule-based systems and that once a system is running and working it can include a large amount of training data and cover a very wide range of different circumstances. The downside is that it may not be easy to choose adequate features and that a reasonable amount of training data is required. The training data also needs to be labeled if a supervised method is to be used. This might be a very time consuming and tedious task if the data set is large. The other downside is that the data needs to be balanced and also be a good representation of the real data which is going to be used and not be too specific. Otherwise there is a risk of overfitting. Overfitting is when the training data is very specific and very specialized to a certain type of data, meaning that it can be very accurate for the training, but highly inaccurate when tested on other data [5].

2.1.2 Evaluation measures

When evaluating the results of NLP problems there are a few standard measures used to measure the results and be able to compare them. Common terms are precision, recall and F-measure [3]. All three of them are expressed as fractions or percentages and are values between 0 and 1. A higher value means better accuracy and a lower value means worse accuracy.

If an evaluation is being done where segments of texts are classified into categories, for each category a true positive (TP) is a segment which was correctly identified as belonging to that category. A false positive (FP) is a segment which was incorrectly categorized to belong in that category. A true negative (TN) is a segment which was correctly identified as not being part of that category. Lastly a false negative (FN) is a segment which was incorrectly classified as not belonging to that category. Precision and recall are calculated as follows [3]:

$$precision = \frac{TP}{TP + FP} \tag{2.1}$$

$$recall = \frac{TP}{TP + FN} \tag{2.2}$$

F-measure is a combination of precision and recall, which takes both into account. The F-measure is calculated as follows [3]:

$$F_\beta = (1 + \beta^2) \times \frac{precision \times recall}{(\beta^2 \times precision) + recall} \tag{2.3}$$

When calculating the F-measure a β-value has to be selected. The β-value is used to weigh the precision and recall and put extra emphasis on the one that you want to prioritize, a β-value larger than 1 will put more emphasis on recall, whilst a β-value smaller than 1 will put more emphasis on precision.

The most commonly used β-value is 1, and then the F-measure formula can be simplified to:

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.4}$$

Sometimes a naive accuracy measure is used, this is however usually not a very good measure since the true negative category is often very large while it has quite low relevance.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.5}$$

For problems where a single answer which is either correct or incorrect, rather than the expected result being multiple points of data, individual results are collected as a binary correct or incorrect value and in this case the standard evaluation measure is a simple accuracy measure where the accuracy is calculated as follows:

$$accuracy = \frac{correct}{correct + incorrect} \tag{2.6}$$

2.1.3 Web environment

Working with NLP in a web environment has both advantages and disadvantages. Web pages are written in HTML, which in itself is a formal language. However the way it is used and the way information is presented will vary greatly between different web sites. The HTML tags themselves can however give clues to the structure of the web site and what is what. Removing the HTML tags themselves is trivial assuming the web page is somewhat conforming to the official web standards.

The difficult part is that data is not necessarily structured in the same way on different web sites [6]. In fact, websites which to a human observer seem the same when rendered in a web browser, could be completely different in how they are built. Some websites will also greatly violate many of the HTML rules and formalities, and in even more cases violate what is considered as "good practice". Many web browsers are very forgiving with poorly designed HTML which means that a website looking and functioning well is no guarantee that it is well designed in terms of HTML use [7].

A typical website will contain more than the specific data one wants to find. Websites will contain navigational elements such as menus, a website header with a logotype and perhaps other information, links to related articles, social media links, advertisements, user comments, a site footer with some legal information and information about the website, and sometimes other elements as well [7]. Finding structured information among all the noise in a typical website is not trivial [4]. Sometimes the web site can give clues through HTML tags and CSS classes, but some websites will just use very generic tags and design patterns, and in this case the extraction can not rely on HTML tags, but rather have to rely on approaches dependant on language.

DOM tree

A Document Object Model (DOM) tree is a graph representation of the HTML tags on a website. Any tag which is placed within the scope of another tag is considered a child node of that tag. Empty tags (tags which do not themselves contain any other tags or text) such as the <br /> tag and non-HTML tokens such as text are considered as leaf nodes in this tree graph. This is described in figure 2.1. Building a DOM tree can be useful to analyze the content of a website, and some algorithms for some NLP problems are based on DOM trees.

Building a DOM tree may be difficult if the web page does not fully conform to HTML standards, but should be possible for all websites which conform to HTML standards reasonably well [7].

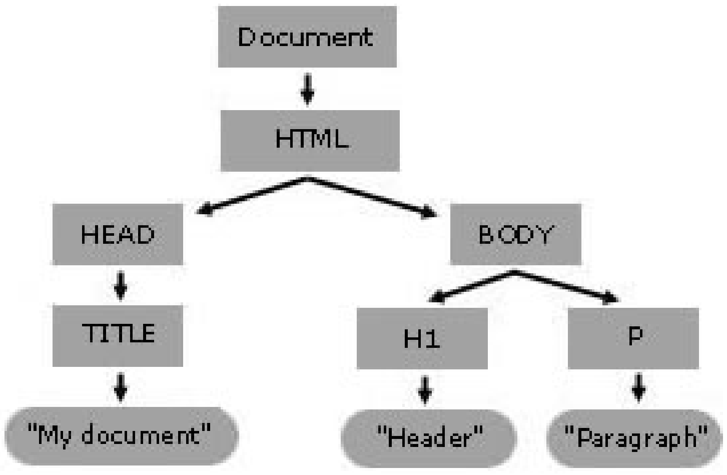

Figure 2.1. An example of a DOM-tree. Image by mozilla.org/David L Baron, used under Creative Commons license.

2.1.4 Stemming

Stemming is the process of reducing a word to its original stem [3]. The stem of the word is the part that has meaning while the rest of the word are parts that put the word in the correct tense, case and form. As an example, the words *apple*, *apples* (plural), *apple's* (possessive singular) and *apples'* (possessive plural) all share the same stem; *apple*.

Sometimes the word can even be derived to an entirely different word class. For example, the word *mash* is the stem of words like *mashing*, *mashable*, *masher* which are all three from different word classes (verb, adjective and noun respectively) but they are all derived from the same stem. Sometimes the stem itself may not be a word, for example *argue*, *arguing* and *argues* have the stem *argu*, which itself is not a word, although it is the stem of several words [3].

Stemming and lemmatization (described below) are often used in NLP subfields such as information retrieval and information extraction. Since in those cases it usually doesn't matter if the word is *tomato* or *tomatoes*, it should be found in an information retrieval system or extracted in an information extraction system regardless of inflected form.

There are several approaches to stemming, most of them deploy a set of simple rules, such as for example if the word ends in an *s*, simply remove that *s*. That deals with all plural cases. Other examples are to remove suffixes such as *-ly*, *-ing* or *-ed* if they are present. Lastly, the stemmer will typically use an exceptions list, a list of words where the other rules do not apply. For example, in cases such as the words *ring* or sing the *-ing* rule should not be applied. There are also cases where special rules need to be used, for example the word *ran* should be identified as the stem *run*, and there are other similar examples of irregular verbs.

A system with similar functionality as this was suggested by Lovins [8] which aside from the already described functionality also applies a recoding procedure to remove a doubled final consonant where applicable, so that for example the word *metallic* is correctly stemmed to *metal* rather than *metall*.

Stemming is a language specific operation and rules need to be crafted separately for different languages. This report focuses on English and Swedish, and even though the specific rules need to be different for English and Swedish, the languages are similar enough that the same rules can be applied, even though the specific suffixes and exceptions list have to be unique for each language.

2.1.5 Lemmatization

Lemmatization is similar to stemming that it is the task to remove any inflected form, but with the difference that lemmatization should leave the lemma, in other words the base form rather than the stem [9]. In the example given in the previous section the lemmatization of the different forms of *apple* are the same as the stemming, however this is not the case in the different derived words from *mash*, since *mashable* and *masher* are themselves lemmas. The lemma of a word can never be of a different word class, and *mashable* and *masher* are adjectives and nouns respectively whilst the stem *mash* is a verb and hence not the lemma of the words.

Unlike stemming, lemmatization will require the use of some form of dictionary, since it is not possible to write a set of rules which will accurately lemmatise all possible words. This makes lemmatization more complex and more resource-demanding than stemming. The results are however often better and more useful, at least in cases where it is important to distinguish between different word classes and when words genuinely have different meanings even though they originate from the same stem.

Just like stemming lemmatization is a language specific operation which needs to applied in a separate way for each language [9].

2.1.6 Tokenisation

Tokenisation is the process of dividing text into separate tokens. A token in this context is usually defined as a word, and splitting a text into an ordered list of words is a relatively trivial task in both English and Swedish. However, there are still cases where more specific cases need to be defined how to be handled. For example, in some cases punctuation can be completely ignored and even removed, in some cases it is included in the nearest token, and in other cases it can be considered a token of its own. There are also cases where the punctuation has a further meaning, in a more informal context the punctuation might be a *smiley*, and it could also be a part of an URL. For example the URL *http://www.google.com/* contains punctuation which is vital to the token (in this case an URL) and should definitely not be removed or considered a token of its own. Another example of this is an HTML tag, which should also be considered as a single token, despite containing spaces and maybe punctuation. A number should also be considered a single token, How numbers are written may differ between languages, for example the number *12,345.67* would be written as *12 345,67* in Swedish.

2.1.7 n-grams

An n-gram is a sequence of *n* tokens, where *n* can be any integer, typically values of *n* between 2 and 4 are used, but other values are also possible. n-grams are usually used in various statistical methods to determine the probability of a certain sequence [3]. In this context, tokens may be either words or characters. *"Hello world"* is a 2-gram on a word level, however if it was to be split into 2-grams on a character level there would be 10 separate 2-grams, one for each possible sequence of 2 characters; *He*, *el*, *ll*, and so on.

2.2 Language Identification

There are multiple approaches to determining which language a text is written in. Most of them utilise n-grams on either a word or character level. The advantage of character n-grams is that systems are smaller and easier to design, they can also be trained on a smaller corpus and still produce results of high quality. They can also capture features from very rare or unknown words and cases where there are spelling mistakes. The downside of character grams is that in a very sophisticated system it may not be enough to distinguish between two very similar languages, and in those cases word grams may be more useful, although they require more training data to work reliably.

Baldwin et al. describe several techniques for identifying language, which have all proved successful in various contexts [10]. They describe several methods which all rely on comparing n-gram frequencies within documents. To compare the different documents they compare them using *cosine similarity*, *skew divergence*, *out-of-place ranking*, *naive bayes* and *support vector machines* (SVM). For the first three of those methods they are also testing two different methods of comparison. Firstly comparing against all training documents individually and finding a nearest neighbor. Secondly, the other method is treating all training documents from a single language as one and finding the most likely match given all data.

The results show that all methods work relatively well. When more languages are tested the cosine similarity comparison is working better, while when the languages tested are fewer and each individual text longer, the overall accuracy is higher and SVM performs the best. The F-1 result for 10 different languages using SVM (the best performing classifier) was 98.7%. For a test of 67 languages with a shorter average length the cosine similarity classifier (the best performing classifier) gave an F-1 result of 86.9% [10]. These results indicate that a classifier will perform better if fewer languages are considered and the texts are longer. Generally all classifiers performed well, which suggests that if this experiment is reduced to just two languages the performance should be very good. English and Swedish are both

Germanic languages, but they are still not very similar considering they are from the same language family. There are some distinctive differences between them and relatively few common words are shared between the two languages.

Another method suggested by Grefenstette [11] also utilized a list of common short words alongside n-gram comparisons. This experiment extracted around 1000-2500 common short words for each language from a corpus alongside with trigrams from the same corpus. The occurrences of these short common words were then compared between the training corpus and test data. The test was conducted on texts of varying length and the results clearly show that this method gets better if the texts are longer. For longer texts (at least 50 words) this method achieved a near 100% accuracy in a test with nine different languages [11].

2.3 Part-of-speech tagging

Part-of-speech tagging is the process of tagging words according to their part of speech category. There are 9 common part of speech categories (noun, verb, adjective, adverb, pronoun, preposition, conjunction, article and interjection) in English, and the amount is typically similar in most European languages. The categories are however often divided into subcategories so the actual amount of different tags can be significantly higher.

Many words are relatively common and can only appear in one category, those words are easy to categorise directly out of a dictionary, but many words can appear in multiple categories and some words are so rare that even a very exhaustive dictionary does not have the word listed. In those specific cases more sophisticated methods have to be applied.

A commonly used part-of-speech tagger is the *Brill tagger*, invented by Eric Brill [12]. The tagger works by initially tagging all known words with the most likely category for that specific word, and then naively tagging all unknown words as nouns as most new words are usually nouns (particularly proper names). The tagger continues by applying a set of rules to change the chosen tags of words according to a set of rules. For example tags are changed if the sequence of tags is impossible, or if specific suffixes of an unknown word suggest it should not be classified as a noun. For instance if the word has the suffix *-ous* it gets tagged as an adjective, since nearly all words ending in *-ous* are adjectives.

Brill achieved a 95% accuracy rate using this method, which is comparable to what other taggers using different methods have achieved [12].

Another typical method to do part-of-speech tagging is to use statistical models and apply them to a Markov chain. An example of this is the work by Brants which uses a Markov chain generated from trigram statistics to create a probabilistic tagger. This tagger scored around 96% accuracy when tested on multiple different corpora [13].

2.4 Named Entity Recognition

Named Entity Recognition (NER) is the problem of locating entities within a text and categorizing them into predefined categories [14]. Common categories used are location, person, organization, date, time and currency. The categoriser has to be able to distinguish between different categories in the cases where multiple categories are possible. “New York Times“ should for example be classified as an organization and not as the location “New York“ followed by the word “Times“. There are other cases where this can be even trickier as even the same set of words may mean different entities, for example “Toyota“ can refer to the car manufacturer but also to the Japanese city with the same name. In some situations a NER system also has the additional task of identifying which of several different entities is referred to in cases where multiple options are available. For example “Anne Hathaway“ may refer to the contemporary American actress, but also to the wife of William Shakespeare. “Birmingham“ may refer to the English city as well as the American city in Alabama.

Entities such as people, locations and organizations are often identified using a predefined dictionary as aid, sometimes this dictionary is known as a *gazetteer*, an old name for a list of places within a map or atlas.

Named Entity Recognition will be used in this study in order to identify the ingredients in a recipe. This is not identical to the standard NER problem, however, similar techniques and rules should be be applicable.

The NER problem was one of the main problems to attempt for the Message Understanding Conferences (MUC) held during the 1990s. The last conference, MUC-7, was held in 1997 and the best performing system achieved an F-1 score of 93.39% [15, 16]. That system was developed by Mikheev et al. [16] for the conference. The system used a rule based algorithm with several steps. It starts by tokenising the text and then applying a set of rules in order to identify as many entities as possible. Initially the text is tagged with a part-of-speech tagger. The entities will typically consist of proper nouns, but some organization names may also consist of common nouns (for example “United Nations“) and some names of locations or people may partly contain common nouns and other word classes (for example “Peter the Great“).

It then goes on by identifying what it refers to as “sure-fire“ cases, where there is a definite match, such as if a title such as Mr. Mrs. or Dr. is used just before a name, or if Ltd. follows after an entity it is identified as an organization. There are also some more complex sure-fire rules, for example the regular expressions seen in table 2.1.

In the examples in table 2.1 X is any uppercase letter, x is any lowercase letter, PROF is a known profession, LOC is a known possible location. The other symbols have their usual meaning with a regular expression, a plus sign means “one or more times“, an asterisk means “zero, one or more times“ and a question mark means

Xxxx+ (is|was) (a|an)? (ADJECTIVE)* PROF	Xxxx+ identified as a person
Xxxx+ (himself|herself)	Xxxx+ identified as a person
in LOC	LOC identified as a location

Table 2.1. An example of some of the regular expressions used.

"zero or one time".

Those rules will identify entities like "Smith is a tall baker" or "I live in Stockholm".

The second step follows a second set of rules where it tags any entities which match those already tagged with the "sure-fire" rules, for example if *Stockholm* was tagged as a location with the rule as described above, all mentions of *Stockholm* within the text are also tagged as locations. Lastly, all entities which have not yet been given a tag are tagged with the tag which is most likely for that specific entity according to the gazetteer used, for example in the previously mentioned example "Toyota" is much more likely to refer to the car manufacturer than the city.

After the first, "sure-fire" step, the precision is very high, over 96% for all three categories, while the recall is much lower, only about 40%. As the algorithm goes over each step the recall increases greatly while the precision drops slightly. At the end, the total precision is 97%, 95% and 93% for people, organizations and locations respectively while the recall for the categories are 95%, 91% and 95% respectively. The combined F-1 value (also including time and money entities) is 93.39% [16].

Named Entity Recognition was also the objective of the CoNLL-2003 shared task with the additional requirement that the system had to be language independent and be able to be applied to multiple languages and not be tailored for a specific language. Four different categories were used; people, locations, organization and other named entities which do not fit into either of those categories. An example of such an entity is for example "Columbia" referring to the space shuttle. It is neither a person, location or organization, yet still a named entity. This meant that machine learning methods had to be used, commonly used methods were the *Maximum Entropy Model* and *Hidden Markov Models* [17].

The best performing system was created by Florian et al. and used a combination of four different methods; a *robust linear classifier*, a *maximum entropy classifier*, a *transformation-based learning classifier* and *hidden markov models* [18].

The features used by the classifiers include the surrounding words, lemmas of the words, prefixes and suffixes of the words and part-of-speech tags (5 word window size), surrounding text chunks as well as gazetteer information. The classifiers were trained on newspaper articles containing around 200,000 tokens and then evaluated on another set of articles containing a total of around 50,000 tokens. The result of the evaluation was an F-1 value of 88.76% for the English test. The same classifiers were also run on a similar set of German texts, and then achieved an F-measure score of 72.41%. The lower scores for the German set was consistent among all participants in the shared task [17]. It was attributed to the German training data being more sparse with more rare words. Also, for the German data the standard method of finding entities by looking for capitalized words will not work as well as it usually does since all nouns are capitalized in German [18].

2.5 Content Extraction

Content extraction is an NLP problem which consists of extracting the part of the document which has the main body of text. This is usually applied to web documents, but can also be applied to other documents. Web documents will usually have a lot of parts in them which are not the main body of text, for example navigational elements, logos, a site footer with some general information and usually links to other content such as for example articles related to the current article.

Content extraction will be needed for developing the algorithm to extract the cooking instructions from a recipe. A recipe will typically not contain any major body of text except the cooking instructions and hence general content extraction is likely to work well for extraction the cooking instructions from a recipe.

There are multiple approaches to this, for example Kohlschütter et al. [19] suggest a method using shallow text features to identify which parts of the page is the actual main body of text, and which parts are *"Boilerplate"* parts. *Boilerplate* parts of a web page are the parts which are not necessarily unique to that specific page, such as menus, article recommendations, and general information that is found on all pages on that specific web site. Their method works by combining several methods, all of them at a shallow level. Firstly, the algorithm checks that the block of text is actually containing complete sentences. This is done by detecting the presence of full stops, exclamation marks and question marks and the amount of words between them. Since this is language independent there is no spell or grammar checking of the sentences, just that it appears to be a collection of complete sentences. It also utilises that the main body of text usually comes in a single block that is typically surrounded by boilerplate elements. A header and some navigational elements will usually come before the main body of text in an HTML document and there will usually be a footer with some legal information and contact details below the main body of text. Lastly, it checks features within the text itself, such as the amount of links and other HTML tags as well as the amount of capitalized words. Boilerplate elements will usually have more capitalized words and links than text elements. Combining these techniques, Kohlschütter et al. [19] achieved an F-1 score for content extraction of 95.1% when defining content into two classes (boilerplate and article text) and 91.1% when defining content into four classes (boilerplate, article text, headline and user comment) [19].

Weninger et al. [20] suggest another approach based on tag ratios. The tag ratio is defined as the amount of HTML tags per line compared to the amount of characters outside of an HTML tag per line. Another algorithm will classify which text is part of the main text body and which is not by using these tag ratios. The report mentions different attempts at this, from simply taking all lines that pass a certain threshold, to a clustering method attempting to find clusters of lines with certain tag ratios and including them as a cluster. The most successful method however is an adaption of both those methods, this method considers both the tag ratio of the line itself and also the change from the previous line, and with that it can create a two-dimensional space where each line is a point in that two-dimensional space. These points in this two dimensional space are then clustered into two separate clusters using some clustering method. The points are depending on their cluster decided to be part of the main text body or be part of boilerplate sections. This method achieved an average F-1 value of 93.93% [20].

Another approach by Zhou et al. [21] is a machine learning approach where the web page is initially split into separate segments, each DOM-element containing text is grouped together with the closest parent element which is displayed as a block (rather than an in-line display) to form a segment. These segments from multiple different web sites are clustered using an unsupervised clustering algorithm. The algorithm uses features including the text length, size and position of the block and well as CSS properties such as colour, line height, and various font properties. These different clusters are then labelled, this can be done either manually or by an algorithm using for example common CSS classes among the elements in that cluster. To match parts of new web pages a SVM classifier was used to classify which cluster best matched a segment from a new unseen website. This mix of clustering and later applying an SVM algorithm makes this a semi-supervised learning method. When this method was evaluated with 4-fold cross validation on a selection of websites where some sites (although not the specific page) were included in the training data the F-1 score achieved using this method was 93.82% [21].

Gupta et al. [22] use an approach for context extraction which is based on building a DOM-tree and then making decisions for each node based on a few criteria. These criteria are similar to those suggested by Kohlshütter et al. [19] and the tag ratio proposed by Weninger et al. [20] in that they take into account the amount of HTML tags (tags which do not themselves form new nodes in the DOM-tree), especially the amount of links and images. It also relies on a predefined list of well-known advertising sites to ease detecting if something is an advert. This work does not however ultimately have the goal to only extract the main body of text, but also extract important headlines and other page-specific content. The main objective is to remove menus, forms and adverts. Another downside of this approach is that it may not always be possible to form a useful DOM-tree from a web page if the web page does not fully conform to web standards. Those factors make this approach a less appealing option.

Some approaches are also used by Pasternak et al. [23] to attempt this problem in a few different ways. They establish a baseline by simply selecting the longest continuous text within a <div> or <p> tag. This naive approach yielded an F-1 score of 63.8%. A second approach used is similar to the tag ratio approach suggested by Weninger et al. [20] as it tries to find the longest sequence with relatively few HTML tags. It works by first stemming all text, and reducing all numbers to a “1“ then assigning all words and symbols a score of +1 and all HTML tags with a score of -3.25. After this locate the sequence of these scores which will generate the highest possible score on this particular website. This sequence is then stripped of any remaining tags. An F-1 score of 90.79% was achieved using this method [23].

To improve on this baseline Pasternak et al. [23] go on to implement a new algorithm with a machine learning approach. The entire web page is tokenised and then using a *Naive Bayes classifier* each token is classified as either “in“ (it is part of the main body of text), or “out“ (it is not part of the main body of text) using a tri-gram model which includes the two tokens which came immediately before as well as the most recent HTML tag which has not been closed. This is a supervised classification meaning that it relies on training data which has been labeled. This is then improved upon by classifying unlabeled data and then using that as further training data. This makes the method semi-supervised. The idea behind it is that the weakness of not having proper labeled data is outweighed by the advantage of being able to use much more data since it does not require labeling that data, which is a tedious and time consuming process. The supervised method achieved an F-1 score of 95.4% while the improved semi-supervised method achieved 97.9% [23].

All these methods have in common that they all remove the content within tags which will never contain any body of text, such as the <style>-tag, the <script>-tag and the <head>-tag.

2.6 Conclusions of background study

The task in this report consists of many smaller tasks that are to be combined. Because there are many smaller tasks they can all be evaluated individually and the results combined for an overall result.

The first stage, language identification, is the simplest task as the previous studies suggest that identification over a specific domain of texts with only two languages considered can have very high accuracy no matter the chosen method. n-gram comparisons are going to be used as the previous work by for example Baldwin et al. [10] suggest this as a well-functioning method.

Secondly, for identifying ingredients as well as other short parts like the cooking time and amount of portions a rule-based system is a good candidate for solving the problem. The rule based systems have had good levels of success and especially for a limited problem where a system can be tailored more towards the specific domain and in the case with ingredients a gazetteer can be used alongside with hand crafted rules.

Lastly, using content extraction to extract the cooking instructions the rule based systems are also the most appealing. Simple algorithms to find the longest continuous string with few HTML elements have proven very successful and a good candidate for moving forward.

There are a few conclusions to draw from this literature study; the chances of achieving a good result are high. Many of the studies cited in this report have achieved relatively high degrees of success. Especially with the work in this study being relatively domain-specific and this system only being developed and evaluated on a specific type of text means the system can be tailored for the specific type of texts, in this case a recipe in HTML format.

Chapter 3

Method

This chapter describes the methods used in this project. Firstly the method for language identification as well as the corpus used is described. This is followed by the main Information Extraction sections which first describes the existing system followed by my proposal for an alternative approach to the Information Extraction problem.

3.1 Language Identification

In order to identify language a method using character n-grams and comparisons using cosine similarity. This method was developed by me in a previous project and refined for this experiment.

3.1.1 Generating the n-gram array

The algorithm starts by creating an n-gram array for each language, in this case English and Swedish.

The array is generated by firstly generating an array of length 50^n filled with zeroes. For each n-gram that occurs in the corpus increase the value at the position in the array corresponding to that n-gram by 1. The position is calculated using the following formula:

$$position = \sum_{i=0}^{n-1}(50^i \times c_i) \tag{3.1}$$

Where i is the character position within the n-gram, n is the gram size and c is the numeric value of the character. Which numeric values to assign each character can be decided in various ways, and should not matter significantly. 50 is chosen since that is as many separate characters that are considered, this means that upper/lowercase letters are not considered separate and only minimal consideration is taken to punctuation. Special characters that exist in Swedish are also included, but no special characters from other languages. English has no special characters and accents, and hence none from English are included. This array has to be generated for each language in the corpus as well as the document to be tested.

For this experiment the n-value was chosen to be 3, as the previous studies suggest that an n-gram value of 3 or 4 is optimal.[10, 11] The lower of those values was chosen since it will be more reliable with a small corpus as well as much cheaper computationally as the array and hence the time complexity increases exponentially when n increases.

The *c*-value is decided in the following way:

0	All whitespace characters
1-26	a-z (both upper and lower case)
27-31	å, ä, ö, ü and é (both upper and lower case)
32-41	The numbers 0-9
42-49	All other characters (Unicode UTF-8 value) modulo 8

3.1.2 Cosine similarity

The two different arrays are then compared using cosine similarity to find the language that most similar to the document which is being tested. Comparison using cosine similarity is simply calculating the cosine value between the array of the document that is tested and all possible languages. The cosine similarity of arrays a and b are calculated using the following formula:

$$\cos\theta = \frac{a \cdot b}{|a||b|} \tag{3.2}$$

This calculation can be done in linear time to the size of the arrays and is calculated using the pseudo code in Algorithm 1.

loa 1. Calculate the cosine similarity between vectors VectorA and VectorB.

```
procedure CosineSimilarity($VectorA, VectorB$)
    $product \leftarrow 0$
    $normA \leftarrow 0$
    $normB \leftarrow 0$
    for all a in VectorA and b in VectorB do
        $product \leftarrow product + a \times b$
        $normA \leftarrow normA + a^2$
        $normB \leftarrow normB + b^2$
    end for
    $norm \leftarrow \sqrt{normA \times normB}$
    $result \leftarrow product/norm$
    return $result$
end procedure
```

3.1.3 Corpus

The training corpus consists of a collection of cooking instructions from various recipes gathered from the Internet. The reason for choosing cooking instructions as the corpus is that cooking instructions mainly is the type of text that will be used in the system and hence the same type of texts should be used for training purposes. This means that context specific words and expressions will be contained within the corpus as well, which should provide a better result. The texts were gathered from major recipe websites in both English and Swedish and in total they consist of approximately 1,900 words (around three and a half pages of text) for each of the two languages. The cooking instructions used to make up the corpus are not part of the texts that are evaluated.

3.2 Information Extraction

This section will describe my method for the various parts of information extraction. All methods chosen are rule-based.

3.2.1 Existing system

There is an existing system in place which the algorithms developed for this experiment will be compared to. This existing system was not originally developed in this experiment, however it has received some improvements and fine-tuned parts of it to work better. This system uses the *General Architecture for Text Engineering* (GATE) suite and a set of rule-based methods. The cooking time, number of portions and cooking instructions are all identified by looking for keywords such as “Ready in:“, “Cooking time:“, “Servings:“ and “Instructions:“ and extracting the text coming after those keywords. The ingredients are identified by defining rules for how an ingredient line can be identified and then extracting all lines which match the identified ingredient lines. An ingredient line will consist of an amount, a unit and an ingredient name matched using a gazetteer. When an ingredient line is matched then other similar HTML elements coming before and after are also matched, this means that if an ingredient line is matched within a <li>-tag with a certain class, then all <li>-tags with the same class in the vicinity will also be classified as ingredient lines. This means that some parts which are not matched by a gazetteer can still be identified. This system is only working in English and not for Swedish.

The results using this system will be compared to the custom algorithms described below.

3.2.2 Gazetteers

For both the existing system and my new algorithms gazetteers are used to identify ingredients. Gazetteers for ingredients in both English and Swedish were created by extracting all ingredients from the Ingredient Matcher database. A gazetteer for measurements in English was created in the same way and a Swedish version was created by manually translating the English measurements into Swedish. Only measurements which will occur in Swedish recipes were included, a total of 82 measurements. A gazetteer with adjectives in English was obtained from the GATE suite and a Swedish gazetteer of adjectives was created by manually translating a small subset of those adjectives which are likely to occur in recipes. In total 726 English adjectives were used and the Swedish subset consisted of just 91 adjectives.

3.2.3 Preprocessing

To ensure that the algorithms can work correctly the input from the webpages and gazetteers has to be preprocessed. The preprocessing steps are tokenisation and reverse stemming.

Tokenisation

The first step of the information extraction part is to tokenise the input from the HTML page. A token has for this case been defined as either an HTML tag, or a word of free text outside of an HTML tag, and a word is defined as a sequence of non-whitespace characters separated by one or more whitespace characters.

The tokeniser initially removes the <script> and <style> tags as well as the content within them as that content should never be matched and may provide very odd tokens that could skew the results. The second part is to match all HTML tags. An HTML tag is in this case defined as starting with a *less than* sign (<) and ending with a *greater than* sign (>). This is not necessarily an absolute definition, as the signs may be used in cases when they are not HTML tags, but this functions well for the purposes of this experiment. Lastly all remaining text is split by whitespace to form the text tokens.

The tokens are labelled as either an HTML-tag token or a text token. This separation is used in multiple parts of the information extraction, either to generate different scores for the two types of tokens for the extraction of cooking instructions as well as being able to retrieve only one type of tokens, since the other parts of the information extraction algorithms use only the text tokens and not the HTML tokens.

Reverse stemming

Reverse stemming is to generate all possible variations of a word from the base word rather than the opposite. This approach was chosen as it is simpler than regular stemming, and that it can easily be applied to the gazetteer in the initialisation process rather than at run-time. This reverse stemming is achieved by allowing any ending to the base form of the nouns and adjectives, meaning that the word *tomato* being present in the gazetteer will allow the word *tomatoes*, but also *tomatoknife* and *tomato-* (with any possible ending). The latter two being allowed doesn't matter significantly as neither of them is very likely to occur, and the recipe import can not be expected to work correctly for websites with significant errors.

The ingredients in a recipe will always be listed in indefinite and non-possessive form and the only variations is for singular and plural. In English words are given their plural form by adding -s or -es to the end of the word. However, for Swedish some words are also reshaped. If a word ends in *-a* then the alternative ending *-or* will also be allowed. This means for example the words *paprikor* (*bell peppers*) and *palsternackor* (*parsnips*) will be included as *paprika* (*bell pepper*) and *palsternacka* (*parsnip*) are. Similarly, if a word has the ending *-ot*, then the ending *-ötter* is also included to allow words such as *morötter* (*carrots*) and *kålrötter* (*Swedish turnips*). The following rules were used for Swedish:

Singular ending	Plural ending
-a	-or
-ot	-ötter
-el	-lar
-e	-ar

This will generate words which are not actual words, but since the incorrect words are extremely unlikely to occur in any recipe, that does not matter since they will never be matched anyway.

3.2.4 Identifying ingredients

The method for identifying ingredients is based on the rule-based NER system developed by Mikheev et al.[16] and uses defined rules to describe how ingredients are written in a recipe. A gazetteer with names of ingredients as well as names of units and adjectives was used to do the matching. Ingredients are then matched using a set of predefined rules which describe how ingredients are typically listed in a recipe. The basic rule can be described as follows:

NUMBER (UNIT)? (ADJECTIVE)* INGREDIENT

Where the number can be any number as either an integer, a decimal number or written as a fraction. The unit and adjectives are both optional (which is indicated by the question marks) and a gazetteer is used to identify them, and lastly the ingredient, which is also matched using a gazetteer. The space between the number and unit is optional.

This rule identifies ingredient lines such as “200 grams ripe bell pepper“, “100ml milk“ and “3 bananas“.

These ingredients are then filtered to attempt to filter out any ingredients which should not be listed, if they are for example written among the user comments, cooking instructions or another part of the text. The goal is to only return ingredients which are written in the ingredient list and not any others.

The ingredient list is usually written separately from other parts of the website and in a consistent order. To identify which ingredients do not belong to the ingredient list, the average token distance between all ingredients is calculated, as well as the median position of the ingredients. Any ingredient that deviates too much from the average distance between ingredients or too much from the distance to median is removed. The cutoff was set at 3.5 times the average distance to the previous and next ingredients, and 2.5 times the average distance from the median ingredient.

These cutoff points were revised a few times before being chosen. Initially they were set lower to keep the limitations on the ingredient list strict, however initial testing showed that increasing the cutoff values to the current values gave better results and the cases of false negatives (i.e. ingredients which were not correctly identified) were fewer. This initial testing was not done with the same recipes as the evaluation. In most cases however the correct ingredients are well within those margins and and the incorrect ingredients are far outside those margins.

Strict ingredients

Strict ingredients are ingredients matched only by their name with the help of a gazetteer, they do not require an amount or a unit. Strict ingredients can be for example oil or butter for frying in or seasoning like for example salt and pepper which are often written without any unit or amount.

These ingredients are filtered in a similar way as the regular ingredients, except the cutoff levels are even lower, 2.5 and 2.1 respectively, and the strict ingredients are not included when calculating the average distances, and only distances to non-strict ingredients are used when calculating distances. Strict ingredients are also unique and any duplicates are removed. This includes ingredients which were also defined as regular ingredients. If *salt* was identified as a regular ingredient it will not be identified as a strict ingredient.

3.2.5 Identifying cooking instructions

The initial method is based on the method developed by Pasternack et al. [23] as well as the tag ratio by Weninger et al. [20] that it uses the ratio of HTML tags compared to text tokens.

Usually on a recipe page there will not be much text body aside from the cooking instructions and perhaps a small description about the dish itself and its origins. Ideally a description which not part of the cooking instructions should not be included.

The algorithm assigns a value to each token, +1.0 is used for text tokens and -2.25 for HTML tokens. The algorithm will then find the sequence which has the highest possible combined score. These values are not identical to those used by Pasternack et al. [23], however early testing results shows better results for using -2.25 than -3.25 as done by Pasternack et al.

This is described in algorithm 2 on page 23.

loa 2. Calculate the start and end positions of the page content.

```
procedure FindPositions($values$)
        ▷ The vector $values$ contains the values of all tokens in the document, in
    order. The values are predefined as either +1 or -2.25.

    $bestScore \leftarrow 0$
    $bestStart \leftarrow 0$
    $bestEnd \leftarrow 0$

    for all positions as startPos do
        $thisScore \leftarrow 0$

        for all positions $\geq$ startPos as endPos do
            $thisScore \leftarrow thisScore + values[endPos]$

            if $thisScore < 0$ then
                break                         ▷ The best score will never be negative.
            end if

            if $thisScore > bestScore$ then
                $bestScore \leftarrow thisScore$
                $bestStart \leftarrow startPos$
                $bestEnd \leftarrow endPos$
            end if

        end for
    end for

    return $bestStart$, $bestEnd$
end procedure
```

3.2.6 Identifying cooking time

To identify cooking time a similar approach as for identifying ingredients. A set of rules describing how time can be expressed is used to extract all possible times. When all possible times have been extracted the most likely time has to be chosen. This is done by once again applying a set of rules. Times in English are matched with the following regular expression:

```
\s+((\d+)\s*(h|hrs?|hours?))?\s*((\d+)\s*(m|mins?|minutes?))?\s+
```

This matches a time consisting of a number, followed by an optional space, then the word “hour“ in some text description (may be abbreviated and/or in plural), another optional space followed by a number and finally the word “minute“ (which may also be abbreviated and/or in plural). Both the hour and minute parts are optional, but obviously at least one of them have to occur. All times are collected and then converted to a time in minutes so all times can be compared as numbers.

The times are then filtered to find the most likely time. If at any point there is just one time remaining that time is chosen. Initially the times which are positioned too far from the ingredients are removed to not include any times which are presented in the navigational menus, user comments or other parts of the web page which are not a part of the recipe. Times which occur in the body of text that is the cooking instructions are given a lower probability, and longer times are given a higher probability, since usually there are many times listed throughout a recipe but the highest time is more likely to be the sum of the time required for each step. Times which are simply too long (more than a couple of hours) are given a lower probability. If a time is preceded by the word “time“, followed by optional colons it is given a higher probability since usually the cooking time is presented as “cooking time“, “preparation time“, “total time“, or simply “time“. The remaining time with the highest probability is chosen as the most likely.

3.2.7 Identifying number of portions

In order to identify the number of portions a simple rule-based approach has been used. Initially all HTML tags are stripped and all excess whitespace is cleaned in order to have a sequence of all text on the page with each word simply separated by a space. In this sequence the algorithm simply searches for the words: *serving*, *serve*, *yield* and *portion* with an adjacent number. Variations of those words such as plural forms as well as an optional colon is also permitted.

The reason for the differentiation between this method and the method for filtering times is that the number of portions simply is a number while the cooking time is a time entity. Hence the number of portions usually has to be clearly labeled as such while the cooking time can sometimes be displayed without a clear label of what the time is referring to.

If multiple matches of possible portion amounts are found then they are filtered in a similar fashion as the cooking times. Portion amounts which are positioned too far from the ingredient list are deemed to not be correct, and if multiple portion amounts remain then the one which is closest to 6 is chosen. The reason for this is that the usual for a recipe is that it yields around 4-6 portions. The upper value of the range, in this case 6, is chosen since a recipe for around 10 people if a dish is more festive or a dessert is more common than a recipe with a single serving, therefore a higher number should be given preference. Numbers which are too high will also be less likely. However, it is entirely possible that a recipe for baking cookies might yield a large amount of cookies, hence large numbers can not be distrusted.

If no match at all is found, a naive guess of four portions is made. Recipes are often designed to make around four portions.

3.3 Experiment

To test these methods a total of 60 recipes will be tested for all different parts, both the language identification system and the information extraction part. All recipes were initially tested in the language identification process. The content extracted by the content extraction algorithm was used for the language identification. This works since even if the content extraction algorithm does not correctly capture the cooking instructions it will still capture a body of text from the webpage and under the assumption that all text on the web pages is in the same language the language identification should still be able to identify the language correctly.

The information extraction part is tested with the language parameter already set, this is because a failure of the language identification should not lead to a failure of the information extraction. Hence the two parts are tested separately, and even though the different parts of the information extraction are evaluated separately they are tested simultaneously since some parts will depend on each other.

3.3.1 Recipes

The 60 recipes were gathered from various recipe websites on the Internet, half of the recipes are in English and half are in Swedish. In total the recipes were gathered from 20 different websites, 10 in each language, and three different recipes were taken from each website.

To choose the recipes, functions to retrieve random recipes were used on websites that had that as a feature, and for the others, recipes suggested as popular recipes were chosen. Consideration was taken to get recipes from a variety of cuisines and categories. A full list of the recipes is available in appendix A.

Chapter 4

Results

The results of the experiments are presented in this section. First the result of the language identification experiment is presented, followed by the results of the Information Extraction experiments. The Information Extraction experiment consists of multiple parts, the ingredient identification, extraction of cooking instructions, identifying the number of portions and the identifying the cooking time.

4.1 Language Identification

The program for identifying language was tested on all 60 recipes described in section 3.3.1 and for each recipe the program answered if English or Swedish was the most likely language that the recipe was written in.

Character 3-grams were used when comparing each document vector to the vector or each of the two languages.

Using those parameters an accuracy result of 100% was achieved, in other words the program correctly identified if the text was written in English or Swedish every single time of the 60 recipes tested. These results are shown in table 4.1.

Language	**Correct**	
English	30/30	100%
Swedish	30/30	100%
Total	60/60	100%

Table 4.1. The results of the language identification experiment.

4.2 Information Extraction

The same 60 recipes from 20 different websites as in the language identification experiment were used for the information extraction part.

The identification of ingredients and cooking instructions are measured in precision, recall and F1-value whilst the cooking time and amount of portions are measured in accuracy as there is only one correct answer and no partial credit.

For identification of cooking instructions the precision and recall values are calculated with each word in the document being considered a true positive, false positive, true negative or false negative depending on its classification.

Some recipes do not specify a cooking time or an amount of portions, these recipes are only included in the ingredient identification and cooking instructions part. If the portion amount is a range (for example 4-6 portions), the integer in the middle of the range (rounded down) should be answered.

The results of the ingredient identification can be found in table 4.2 and the results of the identification of cooking instructions in table 4.3. The tables show the precision, recall and F1 value of each algorithm as well as the normalised precision, recall and F1 values. Normalised values means that each recipe is given equal weight no matter the length of the cooking instructions or amount of of ingredients. For the non-normalised values a recipe with 10 ingredients and cooking instructions with 1,000 characters is considered twice as important as a recipe with 5 ingredients and 500 characters long cooking instructions.

The results show that the results of the existing system as well as the new algorithms for Swedish are very similar, the F1-scores are 88.8% and 88.9% respectively. The results for the new algorithm in English are slightly worse with an F1-score of 83.8%. For the existing system the precision and recall values are closer to each other than for the new algorithms were the gap between the two is larger. The difference was especially large for recall where the existing system performed clearly better although with slightly worse precision. The full results can be seen in table 4.2.

The normalised values are very similar to the non-normalised values, this is an indication that for the ingredient identification there is no notable difference in difficulty to extract the ingredients regardless of how many are listed in a recipe.

Experiment	**Prec**	**Rec**	**F1-value**	**P (norm)**	**R (norm)**	**F1 (norm)**
Existing system	90.4%	87.2%	88.8%	90.2%	88.1%	89.2%
My algorithms (Eng)	86.3%	81.5%	83.8%	88.6%	79.7%	83.9%
My algorithms (Swe)	92.9%	85.3%	88.9%	92.3%	84.9%	88.4%

Table 4.2. The results of the ingredient identification.

For the identification of cooking instructions the differences between the different systems were larger. The new algorithms performed better than the existing system by a large margin. The precision of the existing system was slightly higher, however the recall significantly lower.

The non-normalised scores are also higher than the normalised scores. This indicates that the performance is better when the cooking instructions are longer.

The full results of identification of cooking instructions can be found in table 4.3.

Experiment	**Prec**	**Rec**	**F1-value**	**P (norm)**	**R (norm)**	**F1 (norm)**
Existing system	95.0%	50.9%	66.3%	93.9%	50.0%	65.3%
My algorithms (Eng)	92.4%	96.6%	94.5%	87.3%	89.8%	88.5%
My algorithms (Swe)	78.7%	83.4%	81.0%	78.5%	77.8%	78.1%

Table 4.3. The results for identifying cooking instructions.

The percentages for instructions are dragged down significantly by the fact that for some websites the algorithm fails completely and gets a zero result for both precision and recall. This zero result occurs when the algorithm returns another body of text than the cooking instructions. If the zero values are not included, the F1 value for Swedish is 95.2%, and the recall becomes especially high - 98.1%. The normalised F1 value for Swedish becomes 93.8% with both precision and recall very close to that value.

For English this also leads to a significant improvement, the non-normalised F1 value becomes 97.7% and the normalised value as high as 98.4%. Just as for Swedish, especially the recall is increased significantly, in the non-normalised case to as high as 99.9%.

The results of the identification of cooking time and number of portions show that the new algorithms overall performed better than the existing system. The existing system achieved only 20% correct answers for cooking time while the new algorithms performed significantly better, 88.0% and 79.2% for English and Swedish respectively. For identification of amount of portions the stand out value is the result for the new algorithm in English was as high as 96.4%. Full results can be found in table 4.4.

System	**Cooking time**		**Number of portions**	
Existing system	5/25	20%	22/28	79%
My algorithms (English)	22/25	88%	27/28	96%
My algorithms (Swedish)	19/24	79%	23/30	77%

Table 4.4. The results of identifying cooking time and amount of portions.

For Swedish, 7 out of the 23 correct answers for the new algorithm were achieved by guessing 4 portions when no portion amount was found. By comparison none of the answers for English were given by guessing.

If the existing system does not find a suitable answer it will not make guesses of any kind or answer with for example a time which was found but has no indication of being a cooking time. This means that most of the time the existing system will give a null answer rather than giving an incorrect answer when no cooking time can be found.

The new algorithm runs faster than the existing system. The average run time for the new algorithms is a few hundred milliseconds, compared to a few thousand milliseconds for the existing system. The algorithm is slightly faster for Swedish than for English, due to the English gazetteers being longer and hence the matching takes more time.

Chapter 5

Discussion

This section contains a discussion about the results of the experiments. The chapter begins with a discussion about the experiments in general. This is followed with a discussion about the language identification results. Lastly the majority of the chapter is dedicated to the discussion about the Information Extraction experiments, and each part of the Information Extraction experiment is discussed individually.

5.1 Experiment

The experiment was conducted with only 60 recipes, which is relatively few meaning that just a few outlier results may have an effect on the final results. With 30 recipes from each language and 3 recipes from each website this means that only 10 different websites have been used for each language and hence each website can have a large impact and skew the results. For example for the website *food.com* none of the cooking instructions for any of the 3 recipes was correctly identified.

The recipes themselves are not all randomly selected. If the website had a feature to retrieve random recipes, this feature was used. In the other cases recipes were retrieved by using popular or recommended recipes. There was no conscious bias when choosing the recipes but there may be an unconscious bias or random effects of choosing certain recipes from certain websites which may have skewed the results.

The recipes were all from major and well known recipe websites, but there are other sources of recipes on the Internet, for example through food and recipe blogs or minor websites which could be constructed in different ways. The Internet is also an ever-changing medium and in the future websites may be built using different techniques which might make this method obsolete.

5.2 Language Identification

The language identification solution is working very well, the language of every single recipe was correctly identified as either English or Swedish respectively. The high success may be attributed to the relative ease of the language identification problem, especially since many aspects of this experiment make it an experiment under conditions which in many ways are ideal for language identification.

There are only two languages to distinguish between, and those two are not extremely similar despite being from the same language family. There are even some letters that are unique to each language, Swedish uses the letters å, ä and ö which do not exist at all in English. The texts are domain specific and hence the training data can also be domain specific since the comparisons do not need to work in general but only within the specific domain. The cooking instructions are typically quite long rather than just being one or two short sentences. This means that the data itself is larger which means the risk of an error is smaller.

Overall the combination of these beneficial circumstances as well as the relative ease of the problem makes the language identification results a very good result but also an unsurprising result.

5.3 Information Extraction

The results of the Information Extraction experiment are in general good with the F1-values and accuracies being relatively high. This means that it can be concluded that this method works reasonably well for Information Extraction in this domain. The results are however not accurate enough so that the system can be reliably used without a supervising user to correct errors that may occur.

5.3.1 Ingredient Identification

The new algorithms developed in this experiment were not able to reach performance as high as the existing system and while the precision was comparably high, the recall was lower and dragged down the F1-value. For a system like this which will require a supervising user it is arguably better to have a higher recall than precision if a choice has to be made, as it is easier for a user to spot an incorrect ingredient and remove it than to include a missing ingredient.

The reason for the existing system functioning better than the new algorithm is likely to be the new algorithm's large dependence on gazetteers. Even a very exhaustive gazetteer will never be complete enough to cover all possibilities. Hence it is not surprising that the existing system which does not fully rely on gazetteers provides a better result. The higher precision for the new system is most likely also due to the high reliance on gazetteers. If every ingredient is matched by a gazetteer then the risk of false positives is smaller. The reason why false positives still occur is because ingredients like *mashed tomatoes* may be matched as just *tomatoes* and for example *rice paper* be matched as simply *rice*.

The new algorithm performed better for Swedish than for English. This may be due to that Swedish recipes are not as varied as English recipes and that the ingredients are more common. Swedish recipes are typically only aimed towards users in Sweden and ingredients that can be acquired at Swedish stores, while recipes in English can be aimed at both American or British audiences as well as international audiences. Building a gazetteer for those cases would be more difficult as more ingredients would have to be included.

Swedish recipes usually also included more concise descriptions of ingredients with fewer adjectives or other descriptive words. As an example an English recipe might read “two lovely sprigs of fresh thyme“ while a Swedish recipe might read “2 st timjankvistar“ (literally, *2 sprigs of thyme*). Those may be extreme examples, but cases such as that are contributing to the better results for Swedish than for English.

The algorithm’s performance with an F1-score of 83.8% and 88.9% for English and Swedish respectively was slightly worse than the state-of-the-art systems developed by Mikheev et al. [16] which achieved an F1-score of 93.1%. It is however comparable to the English results achieved by Florian et al. [18] at 88.76% and better than the German results of 72.41%.

The ingredient identification is not an identical problem to the standard NER problem. Ingredients are not capitalised like proper nouns usually are and the standard rules for identifying an entity can not be used. It can also in some ways be easier than the normal NER problem. Gazetteers can be used to a larger extent than for the normal NER problem, and there is only one category to place all entities in rather than the usual NER problem of defining entities into different categories. For a recipe it is also important that ingredients are only identified once, and not for example once in the ingredient list and then again in the cooking instructions. Simply removing all duplicates is not an adequate solution since some recipes might contain duplicate ingredients if for example the ingredients for the main dish and the sauce are listed separately.

5.3.2 Extraction of Cooking Instructions

The new algorithm performed significantly better than the existing system for extraction of cooking instructions. The new algorithm working with general content extraction principles rather than looking for specific keywords has provided significant improvement. The precision is slightly lower for the new algorithm however the recall is significantly higher.

Sacrificing some precision to gain recall is a common dilemma, and in this case the precision sacrifice is relatively small for a large increase in recall. For the English recipes the precision was only dropped from 95.0% to 92.4% while the recall was increased to 96.6% from 50.9% by using the new algorithm compared to the existing system.

The performance of the algorithm was comparable to the results of other content extraction systems. The F1-score of 94.5% for English is comparable to 95.1% by Kohlshütter et al. [19], 93.93% by Weninger et al. [20] and 97.9% by Pasternak et al. [23]

Content extraction from recipes is not necessarily easier or harder than extraction from news articles which was the most common text type used for the previous studies. Recipes usually have the cooking instructions as the main body of text and there are usually not any other major parts of text, even though user comments, page headers and footers may occur. This is similar to how news articles are usually presented on web pages.

For all values, but particularly the English recipes with the new algorithm, the normalised values are lower than the non-normalised. This means that it can be concluded that extracting the cooking instructions successfully is easier when the instructions are longer. This is perhaps not extremely surprising since matching a longer text is easier for the algorithm and reduces the risk of identifying another body of text instead.

The times when the algorithm fails completely are also more frequently occurring when the cooking instructions are relatively short. The algorithm tries to capture the longest reasonably continuous body of text and if the cooking instructions are very short, or for other reasons are not very continuous the risk is greater that another body of text is larger, especially if that text is comparable in length to the cooking instructions.

If the complete failures are not included in the results, the F1-score increases significantly, for the English texts to as high as 97.7%, with especially the recall increasing significantly. This means that most likely it is the specific outlier cases which drag the performance down, and for the average website the algorithm works very well and produces very good results.

The other texts that are sometimes extracted instead of the cooking instructions are texts with legal information about the website, other general information about the website, and sometimes text describing the dish itself rather than the cooking instructions.

Texts incorrectly extracted together with the cooking instructions included nutritional information, and sometimes the ingredient list. Those erroneous extractions lead to a reduction in the precision without affecting recall.

The new algorithm performed better for English than for Swedish. There are no obvious reasons for why this would be the case except that the English and Swedish websites may be built differently. Several of the Swedish websites had some additional information such as nutritional information, allergy information, and information about portion sizes together with the cooking instructions and hence those bits were also extracted which is a contributing factor to the slightly worse results for Swedish.

5.3.3 Cooking time and number of portions

The identification of cooking time and amount of portions generally worked better for English than for Swedish, the difference is especially clear in the identification of number of portions. There are some possible explanations for this. One in particular is that on Swedish websites the portions and also to an extent cooking time were indicated by an icon or small image rather than a word. The Swedish websites also to a larger extent had support for variable portion amounts with for example a dropdown menu or sliding bar which also made detection more difficult since the portion amount is not printed on the webpage in those cases.

When no matching portion amount was found, the algorithm guessed 4 portions. This occurred for 7 recipes in Swedish and none for English. All the 7 guesses for Swedish were correct, and without those guesses the algorithm would only have answered correctly for 16 out of 30 recipes - just over 50% accuracy. The same system for English answered correctly 27 out of 28 times, 96.4% of the time. This means that it can be concluded using keywords and adjacent numbers to identify the number of portions works well for English, but not very well for the Swedish recipes.

For cooking time the new algorithm worked significantly better than the existing algorithm, both for English and for Swedish. Just as for the number of portions the English results are slightly better than the Swedish results. Similar to the number of portions the reason may be that in Swedish the cooking time was often indicated by an icon rather than text.

The algorithm to identify cooking time will give priority to, but not require that a time has an adjacent keyword to indicate that it is a cooking time. The permissive algorithm has resulted in that many cooking times were identified, and the cooking time may be identified even if it is not labeled by a keyword.

The failures in identification of cooking time occur when either time is not explicitly labeled as cooking time, and in the highest time on the page is chosen. In those cases the highest time is usually correct, but sometimes it may not be. Examples are if another time is presented on another part of the page, a link to a related recipe might read “lasagna in under 45 minutes“, a category might be called “under 2 hours“ or it might read “store in the fridge for 3 hours“ within the recipe even though the cooking time is lower than 3 hours.

When the existing systems fails it usually provides a null answer rather than an incorrect answer. This means that the failure of the algorithm will be clear and the user can easily notice that no value was chosen. No value is generally better than an incorrect value. However in this case the existing system performs significantly worse than the new algorithm and in a few cases also answers incorrectly, suggesting that the new algorithms are a clear improvement.

Chapter 6

Conclusions

In this study an F1-score of 83.8% and 88.9% was achieved for ingredient identification for English and Swedish respectively. For identification of cooking instructions 94.5% and 81.0% was achieved for English and Swedish respectively. For cooking time and number of portions the accuracies were 88.0% and 96.4% for English and 79.2% and 76.7% for Swedish. These results are relatively high and in many cases comparable to previous studies and hence it can be concluded that the experiments have been successful.

Even though the results are high, they are not high enough that the system can be fully trusted and used to extract recipes without human supervision. Requiring a human to supervise and correct errors is a weakness of this system, however using this system to extract information from recipes will be significantly quicker than a user manually doing the same work. This is the case even if the time spent to correct errors is accounted for.

The scientific question asked if a rule-based system can be successfully used to identify the different sections which make up a recipe. Based on these results it can be concluded that the system developed here was reasonably successful and hence that rule-based system can be used, even though the results achieved in this experiment were not flawless.

The language identification algorithm worked very well and scored a full 100% accuracy and was hence very successful. Due to this it can be certain that the language identification algorithm will continue to provide good results even in future tests.

The very successful result despite using a relatively simple method is a strong indication that the language identification problem is not very difficult, especially when reduced to just two relatively distinct languages as well as relatively long domain specific texts.

The new algorithms were a clear improvement over the existing system for all parts except the ingredient identification where the results were comparable or slightly worse. This means that adopting the new algorithms for at least the parts where the improvements were clear will provide an overall improvement to the system.

The new algorithms also run significantly quicker than the existing system due to less overhead and less reliance on external libraries. The lower execution time is also an advantage since a slower algorithm will be more annoying for users and put more load on the server.

The existing system will to a greater extent than the new algorithms provide a null answer rather than providing an incorrect answer. This is a strength of the existing system since it is easier to notice and correct a null answer than an incorrect answer. However the overall performance provided an improvement which is significant enough to overlook this flaw.

6.1 Future work

Future work in this area can expand on these algorithms and attempt to provide a system which is working well enough to be used without human supervision. This system may require more complex rules to be able to identify all different parts correctly, and for the very popular recipe websites more specific systems which are specifically tailored for that website can be used. A system which is tailored for a specific website will never be able to work for a general case and may be obsolete if the website is redesigned.

Future work may also use different methods than rule-based methods. For example a machine learning algorithm or a hybrid approach may be used to improve the results, this would however be relatively difficult. Choosing features and algorithms is not trivial for a problem like this.

References

[1] Axel Afzelius and Ragnar Gyllenswärd. "Matrecept". In: *Nytt Juridiskt Arkiv* (1946), p. 371.

[2] United States Copyright Office. *U.S. Copyright Office - Recipes*. 2011. URL: `http://www.copyright.gov/fls/fl122.html` (visited on 05/13/2016).

[3] Dan Jurafsky and James Martin. *Speech and Language Processing*. Pearson Prentice Hall, 2009. ISBN: 9780131873216.

[4] Jim Cowie and Wendy Lehnert. "Information extraction". In: *Communications of the ACM* 39.1 (1996), pp. 80–91.

[5] Laura Chiticariu, Yunyao Li, and Frederick R Reiss. "Rule-Based Information Extraction is Dead! Long Live Rule-Based Information Extraction Systems!" In: *EMNLP*. October. 2013, pp. 827–832.

[6] Raymond Kosala and Hendrik Blockeel. "Web mining research: A survey". In: *ACM SIGKDD Explorations Newsletter* 2.1 (2000), pp. 1–15.

[7] Suhit Gupta et al. "DOM-based content extraction of HTML documents". In: *Proceedings of the 12th international conference on World Wide Web*. ACM. 2003, pp. 207–214.

[8] Julie B Lovins. *Development of a stemming algorithm*. MIT Information Processing Group, Electronic Systems Laboratory Cambridge, 1968.

[9] Joël Plisson, Nada Lavrac, Dunja Mladenic, et al. "A rule based approach to word lemmatization". In: *Proceedings of IS-2004* (2004), pp. 83–86.

[10] Timothy Baldwin and Marco Lui. "Language identification: The long and the short of the matter". In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics. 2010, pp. 229–237.

[11] Gregory Grefenstette. "Comparing two language identification schemes". In: *Proceedings of the 3rd International Conference on the Statistical Analysis of Textual Data (JADT 1995)*. 1995.

[12] Eric Brill. "A simple rule-based part of speech tagger". In: *Proceedings of the workshop on Speech and Natural Language*. Association for Computational Linguistics. 1992, pp. 112–116.

[13] Thorsten Brants. “TnT: a statistical part-of-speech tagger”. In: *Proceedings of the sixth conference on Applied natural language processing*. Association for Computational Linguistics. 2000, pp. 224–231.

[14] David Nadeau and Satoshi Sekine. “A survey of named entity recognition and classification”. In: *Lingvisticae Investigationes* 30.1 (2007), pp. 3–26.

[15] Elaine Marsh and Dennis Perzanowski. “MUC-7 evaluation of IE technology: Overview of results”. In: *Proceedings of the seventh message understanding conference (MUC-7)*. Vol. 20. 1998.

[16] Andrei Mikheev, Claire Grover, and Marc Moens. “Description of the LTG system used for MUC-7”. In: *Proceedings of 7th Message Understanding Conference (MUC-7)*. Fairfax, VA. 1998, pp. 1–12.

[17] Erik F Tjong Kim Sang and Fien De Meulder. “Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition”. In: *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics. 2003, pp. 142–147.

[18] Radu Florian et al. “Named entity recognition through classifier combination”. In: *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*. Association for Computational Linguistics. 2003, pp. 168–171.

[19] Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. “Boilerplate detection using shallow text features”. In: *Proceedings of the third ACM international conference on Web search and data mining*. ACM. 2010, pp. 441–450.

[20] Tim Weninger, William H Hsu, and Jiawei Han. “CETR: content extraction via tag ratios”. In: *Proceedings of the 19th international conference on World wide web*. ACM. 2010, pp. 971–980.

[21] Ziyan Zhou and Muntasir Mashuq. “Web Content Extraction Through Machine Learning”. 2014.

[22] Suhit Gupta et al. “Automating content extraction of html documents”. In: *World Wide Web* 8.2 (2005), pp. 179–224.

[23] Jeff Pasternack and Dan Roth. “Extracting article text from the web with maximum subsequence segmentation”. In: *Proceedings of the 18th international conference on World wide web*. ACM. 2009, pp. 971–980.

Appendix A

Recipes

This appendix contains all recipes used in the experiments. They were all accessed between 26 April and 5 May 2016.

A.1 English

```
http://www.food.com/recipe/tax-time-stroganoff-53075
http://www.food.com/recipe/bourbon-chicken-45809
http://www.food.com/recipe/ww-0-point-weight-watchers-cabbage-soup-128956

http://allrecipes.com/recipe/50870/bbq-teriyaki-pork-kabobs/
http://allrecipes.com/recipe/20156/clone-of-a-cinnabon/
http://allrecipes.com/recipe/85740/cheddar-baked-chicken/

http://www.bbcgoodfood.com/recipes/grilled-sea-trout-raw-pea-salsa
http://www.bbcgoodfood.com/recipes/3091678/vegetable-vegan-biriyani-with-carrot-salad
http://www.bbcgoodfood.com/recipes/rich-ragu

http://www.foodnetwork.com/recipes/banana-bread-recipe2.html
http://www.foodnetwork.com/recipes/rachael-ray/green-salad-with-strawberry-balsamic-vinaigrette-recipe.html
http://www.foodnetwork.com/recipes/food-network-kitchens/pasta-primavera-recipe.html

http://www.jamieoliver.com/recipes/pasta-recipes/spaghetti-atterrati/
http://www.jamieoliver.com/recipes/rice-recipes/yellow-bean-vodka-and-smoked-haddock-risotto/
http://www.jamieoliver.com/recipes/vegetables-recipes/cypriot-style-potato-salad/

http://www.foodandwine.com/recipes/jacques-pepins-favorite-pound-cake
http://www.foodandwine.com/recipes/curried-carrot-and-apple-soup
http://www.foodandwine.com/recipes/grilled-halloumi-and-lentil-salad

http://www.realsimple.com/food-recipes/browse-all-recipes/turkey-meatloaf-mashed-potatoes
http://www.realsimple.com/food-recipes/browse-all-recipes/glazed-pork-and-pineapple-kebabs-0
http://www.realsimple.com/food-recipes/browse-all-recipes/matzo-brei-recipe

http://www.myrecipes.com/recipe/san-antonio-beef-puffy-tacos
http://www.myrecipes.com/recipe/cranberry-sweet-potato-quick-bread
http://www.myrecipes.com/recipe/senegalese-lemon-chicken

https://realfood.tesco.com/recipes/spring-salad-platter.html
https://realfood.tesco.com/recipes/reuben-steak-sandwich-with-skin-on-fries.html
https://realfood.tesco.com/recipes/chocolate-and-avocado-mousse.html

http://www.marthastewart.com/312429/grilled-polenta-and-balsamic-mushrooms
http://www.marthastewart.com/312597/salmon-with-olive-relish
http://www.marthastewart.com/254940/double-diablo-cake
```

A.2 Swedish

```
http://www.koket.se/asiatisk-laxtartar-i-friterat-rispapper
http://www.koket.se/tina-nordstrom/soppor-och-grytor/kott/pulled-pork-pa-tinas-vis/
http://www.koket.se/lomelinos-tartor/linda-lomelino/tarta-med-apelsin-och-dulce-de-leche/

http://www.recepten.se/recept/rabarberkolapaj.html
http://www.recepten.se/recept/paaskgodis.html
http://www.recepten.se/recept/panerad_roedspaetta.html

http://www.ica.se/recept/pannkakssmet-grundrecept-till-plattar-och-pannkakor-292828/
http://www.ica.se/recept/flygande-jacob-720361/
http://www.ica.se/recept/gron-gratang-med-cheddar-och-kavringkrisp-718361/

http://mittkok.expressen.se/recept/lax-citronspett-med-dragonremoulad/
http://mittkok.expressen.se/recept/pumpa-getostpaj/
http://mittkok.expressen.se/recept/ceviche-5/

http://www.tasteline.com/recept/chiapudding-med-granatapple-och-mandelmjolk/
http://www.tasteline.com/recept/mias-melanzane/
http://www.tasteline.com/recept/spanska-kottbullar-med-mandelvinagrett-och-tomatsas/

http://www.arla.se/recept/carbonaragratang/
http://www.arla.se/recept/pulled-pork-med-whisky/
http://www.arla.se/recept/falafel-med-myntadressing/

http://recept.se/content/baconlindade-kycklingfileer-med-quinoasallad
http://recept.se/content/farskostfyllda-kalvfarsburgare-med-rod-pesto-och-rucolasallad
http://recept.se/recept/kycklingfileer-i-ugn-med-vitlok-buljongtarning-och-gradde

https://www.coop.se/Recept--mat/Recept/n/nudelsallad-med-sojaboenor/
https://www.coop.se/Recept--mat/Recept/f/falafelbarens-friterade-aubergine/
https://www.coop.se/Recept--mat/Recept/c/chili-con-carne/

https://www.mathem.se/recept/grillad-lax-med-sparris-och-vasterbottenssas
https://www.mathem.se/recept/nachotallrik-de-lux
https://www.mathem.se/recept/couscoussallad-med-salami--mozzarella-och-persiljefrast-gront

https://www.mat.se/recept/graddig-pasta-med-svamp-2
https://www.mat.se/recept/shepherds-pie-3
https://www.mat.se/recept/ungersk-gulasch
```