Information Extraction - Automatic Recipe Introduction According to an Ontology

Telmo Machado

IST – Instituto Superior Técnico
L^2F – Spoken Language Systems Laboratory – INESC ID Lisboa
Rua Alves Redol 9, 1000-029 Lisboa, Portugal
{trma}@l2f.inesc-id.pt

Abstract. The importance of information extraction has risen lately, as a consequence of the growing available information and it's lack of treatment. Therefore, information extraction systems appear as a mechanism to filter that information and only obtain the needed information, depending on the domain.
This document shows an information extraction system in a specific domain, the cooking domain, which is composed of three modules, preprocessing, recipe processing and output transformation. The main objectives are identifying ingredients and their associated quantities, and identifying the different tasks needed to the preparation phase of each recipe, as well as the needed utensils and the used ingredients in each of these tasks. After identifying the ingredients and the tasks of each recipe, these are placed in a database created for this system that has as base an ontology, which contains a description of the concepts used in this recipe domain.

1 Introduction

Information extraction is a process that searches texts trying to find some relevant information in a specific domain and extracting entities, relations and events from it. Natural language texts usually are non or partially structured, and the main function of information extraction systems is to get the important information from those texts and produce structured information, such as a logical representation (predicates) or a relational database, where the obtained information is introduced. Information extraction is getting more important throughout the time, mainly because of the increase of available information and the need to have access only to the wanted information. Information extraction systems exist for these reasons, and to get only the most important information for a specific domain. Several techniques have been used by these kind of systems, such as pattern discovery and rule based (the most used), syntactic analysis, and learning based, using statistical methods.

The system that is presented here uses mainly syntactic analysis, pattern discovery and is rule based. It reads a recipe from a file, converts the recipe to a normalized format, does a syntactic analysis to each part of the recipe (title, ingredients and preparation) and introduces the extracted data to a domain specific database.

2 Ontology

All this work is based in an ontology [1] created to have some stored knowledge about the recipes domain. This ontology defines the main concepts of the recipe domain, such as food, actions and tasks, and utensils, giving examples of each concept, like “lombo de vaca” (beef sirloin) as food, “ADICIONAR” (add) as action and “fogão” (stove) as utensil.

Food is organized in four main concepts, food (that consists in a food hierarchy of food concepts), solid part food (part of food which is strutured in a hierarchy), food state (characterizes food in four dimensions, alimentary, physical, culinary and segmentation state) and food with state (association between food and food state concepts).

Utensil concept contains a hierarchy of kitchen utensils organized by the action performed by each utensil. They are divided in three principal groups, “Utensílio Manual” (manual utensil), “Electrodoméstico” (appliance) and, the last, “Acessório” (accessory).

The action concept consists of operations that can be executed with utensils and food. It is strutured in three main concepts: simple action, ordered action and complex action. Simple actions define the simplest things done when cooking, like breaking an egg or peeling an apple. There are three types of simple actions, simple actions for food, simple actions for utensils and simple actions for waiting, determined by what is needed for each action (food, utensils or waiting time).

Ordered actions associate a number with any kind of action so that it is possible to define a sequence. Complex actions define more complex tasks performed when cooking, like roasting or frying. Complex actions are executed on food and utensils and define a process, composed by a sequence of ordered actions.

3 System Architecture

The developed system is composed of three modules, pre-processing, recipe processing and output transformation, each one having distinct functions and objectives, and using XIP [2, 3] to do syntactic analysis and extraction of the dependencies. These three modules are presented in the next sections.

3.1 Pre-Processing Module

Pre-processing phase reads a file with a recipe and does some modifications to it. The objective of these modifications is to make the analysis of the second module easier. The main modification consists of adding a period (“.”) to the end of each line in the file, except if it already has one. This is done because XIP (used in the second module) processes a text ”‘phrase by phrase”’, being the period considered a delimiter of phrases. Another modification consists in changing some words that are not correct, like the measure unit "gram" which is incorrectly written as “gr” or “grs”. In this situation, the words are transformed to the correct word (“g”). This is done using the “sed” command, that makes substitutions based on regular expressions. Each recipe is then divided in it's three basic parts, title, ingredients and preparation, creating a file for each part. In between, there should be an empty line so that the division can be made, and this should be the only empty line in the file. This separation is important so that the system knows which part it will process in the remaining modules.

3.2 Recipe Processing

This module is responsible for extracting the information needed for this domain. It uses XIP to do syntactic analysis of the text and to get the dependencies with the information needed about ingredients and actions.

A lexicon is then created that contains food, food states, food parts and utensils. This lexicon does an association between a feature and a word, which means that a word belongs to the category specified by the feature, i.e., if a word has the feature “alimento” (food) then that word is food. The main source of words for this lexicon is the ontology, which has many concepts in the cooking domain.

The next sections show how dependencies are extracted in XIP, and how the ingredients and the actions are extracted from the recipes.

3.3 Syntactic Analysis and Dependency Extraction

XIP [2, 3] is responsible for doing syntactic analysis from the input text and extracting relevant information in form of dependencies.

Dependencies are relations linking two or more words or segments of the text. It is possible to get them through rules, that have three parts:

- Context
- Condition
- Extraction

Context is a regular expression applied to the input text that has to be matched so that the rule can be used. Condition is a boolean condition on dependencies or on a comparison of features associated with words. Extraction is a list of dependencies extracted if the contextual description and the condition are true. For instance, the following rule establishes a CDIR dependency between the heads of the verbal phrase and the nominal phrase.

```
| #2[verb:+];sc{?*,#2[verb:+]}, (ADVP), NP#3 |
 if ( HEAD(#1,#2) & HEAD(#4,#3) & VDOMAIN(?,#1) & ~CDIR(#1,?) &
      ~PREDSUBJ(?,#4) )
   CDIR[post=+](#1,#4)
```

The first line corresponds to context and describe a word that is a verb or a sentence clause chunk terminated by a verb (marking the verb with the variable #2), followed by an optional adverbial phrase and by a nominal chunk marked with variable #3. In the condition are extracted the heads of the marked variables #2 and #3. is checked if there is any VDOMAIN dependency in the phrase analised, and is verified if a CDIR and PREDSUBJ dependencies do not exist. Finally, if context and condition are verified, a dependency CDIR is created.

Ingredients Processing The objective of this step is to extract some dependencies that have information about the ingredients part of the recipes. That dependencies are "INGR" (that tells the food, it's quantity and the unit of measure), "PARTEDE" (represents a connection between a food part and it's corresponding food) and "ESTADO" (tells the state in which the food is). For instance, in the phrase "200 g de bife do lombo de vaca", the dependendies extracted are INGR(200, g, vaca), PARTEDE(bife, vaca) and PARTEDE(lombo, vaca).

There are eigthteen patterns that can be applied in rules to extract these three dependencies that contain the information about the ingredients. This number is so high for such a simple thing because all combinations are considered that exist between food part (it can have 1 or 2 food parts), food, quantitity (it can or cannot be present, or can be a unit food quantity) and some other slight variations.

For example, one of the patterns is <Número> <quantidade> de (<parte de alimento>)+ de <alimento>, which can handle phrases like "200 g de bife do lombo de vaca" and "200 g de lombo de vaca". The rule that apllies to this pattern is

```
| #4NP |
 if( HEAD(#1,#4) & QUANTD(#1[meas:+], #2) &
     MOD(#1, #3[alimento:+]) )
  INGR(#2, #1, #3)
```

that can extract the dependency INGR(200, g, vaca). The context is composed by a single nominal phrase (NP), and then in the condition is extracted the head of that NP (in HEAD dependency) to the variable #1 and is verified if the variable #1 is a measure (in this example, #1 is "g"), if a QUANTD relation between #1 and #2 (#2 is "200") exists and if there is a MOD relation between #1 and a word that is a food (annotated with feature "alimento"). If both context and condition are verified, the dependency written above is extracted.

The rules for the remaining patterns are similar to the one presented here.

Actions Processing In this step, dependencies related to actions are extracted, in a total of 112 actions described in the ontology. The actions are divided in types, using the type of their arguments to do the division. With this division it is possible to get ten types of actions, related with the characteristics of each depedency, i.e., a dependency can have one or more food, one or more utensils, food and utensils, and relations between temperature and food or utensils.

Having in consideration the phrase “Junte os alhos e as cebolas”, which belongs to the type of actions that have more than one food, it is possible to get the dependencies JUNTAR(alhos, e) and JUNTAR(cebolas, e). Below is presented the rule that extracts the dependency JUNTAR for this type of action.

```
| #1VERB[lemma:"juntar"] |
 if( HEAD(#2,#1) &
     (CDIR(#2,#3[alimento:+])|CDIR(#2,#3[partealimento:+])) &
     COORD(#4,#3) )
  JUNTAR(#3,#4)
```

The context contains a verb with the lemma “juntar” (add), the condition is verified if that verb has a CDIR relation with a word that is either a food (“alimento” feature) or a food part (“partealimento” feature), and if there is a relation COORD (coordenation) associated with the food or food part. After these verifications the dependencies above are created.

The remaining nine types of actions have their own rules in order to have the correspondent dependencies extracted.

3.4 Output Transformation

The last module of the system reads the output given by XIP in XML format, that contains all the information about each phrase, including the dependencies extracted, and transforms it in SQL code so that the information about a recipe can be introduced in a database, using Java and it’s mechanisms to manipulate XML and databases.

This module is composed of three components, a state converter, a XML parser and a database connection, which are controlled by two main programs, one for ingredients and other for actions.

The state converter corrects the states (extracted in the dependency “ESTADO”) that are incorrect because XIP is used with an option to show only the lemmas of the words. These states are only those that are annotated by XIP with the past participle grammatical category, because the lemma of words in that category are different of the states previously introduced in the database, and of the surface word (E.g. “assada” has lemma “assar”). So, this module changes the state (which is in lemma form) to the correct state (the surface form which is the same state that is in the database).

The XML parser reads the XML file that results from the previous module and gets only the elements that correspond to the dependencies related to the recipe part that the system is analysing, i.e., when the system is processing the ingredients part, this parser only gets the elements with the dependencies “INGR”, “PARTEDE” and “ESTADO”. When the system processes the preparation module, the parser gets the elements with the dependencies related to all actions plus “PARTEDE” and “ESTADO” dependencies. This component is also responsible for the association to each argument of the dependencies an attribute that tells if that argument is a food or an utensil. This information about the type of the arguments is available in the XML file, and is useful to specify what action is being processed. This component has one more function, to reconstruct food names that are composed of food parts, because only the food is extracted in action and ingredient dependencies. This is done by doing an association between the food name present in the argument of the action or ingredient dependency and the "PARTEDE" dependencies related with that dependency.

The database connection component manages all connections to the database, including opening and closing connections, and executing queries. It also builds all the SQL code needed to introduce the recipes in the database. The set of steps needed to build the SQL code for ingredientes are:

- 1: Get food id;
- 2: Get food with state id;
- 3: Insert food states;
- 4: Get measure id;
- 5: Create query;
- 6: Insert ingredient.

The set of steps for actions are:

- 1: Get action id;
- 2: Get utensil id;
- 3: Get measure id (time or temperature);
- 4: Insert task;
- 5: For each food in the action:
  - 5.1: Get food id;
  - 5.2: Get food with state id;
  - 5.3: Insert food states;
  - 5.4: Associate the task to the food with state.
- 6. Insert ordered task.

Depending on the type of dependency that is handled, the component does or does not apply some steps in order to build the SQL code for insertion in the database. This only applies for actions, because not all actions have utensils, measures or food, like actions for waiting, that doesn't have utensils and food and will not do steps 2 and 5.

The main program manages the interaction with these three components and finds the type of dependency that is processed. This distinction is important for the database connection component because it has different functions based on the type of dependency, handling different types in different ways. For the ingredients part, the distinction is made based on the number of arguments of "INGR" dependency, i.e., if the program gets a dependency "INGR" with one argument, it automatically knows that its argument is a food (E.g. INGR(leite)). The distinction in the actions part is more complex, based on the attributes added in the XML parser which tells if an argument of the dependency is either a food name or an utensil, and that corresponds to the ten types of actions presented in section 3.3. After the program knows what type of dependency it's handling, it prepares the information needed (such as a list of food, a list of states or the utensil needed) and calls the corresponding function in the database connection component so that the SQL code can be created.

3.5 Example of Output

When the system processes a recipe that has the ingredient “200 g de açúcar”, the second module extracts only from that phrase the dependency INGR(200, g, açúcar). From this dependency, the third module creates the SQL so that the ingredient can be introduced in the database. The following SQL code results from the dependency extracted.

```
INSERT INTO Ingrediente(IDReceita, IDMedida, IDAlComEstado, nome)
       VALUES("2", "24", "11", "200 g de açúcar")
```

This code inserts in the “Ingrediente” table the ingredient, which consists of foreign keys to other tables (recipe, measure and food with state tables, respectively). These keys are obtained asking those tables for the corresponding values, “200 g” in measure table and “açúcar” in food with state table.

If the recipe has the action “Junte as cebolas”, the dependency extracted is JUNTAR(cebolas) and the SQL created is the following.

```
INSERT INTO Tarefa(IDAccao, IDUtensilio1, IDUtensilio2, IDTemp)
       VALUES("73", "0","0","0")

INSERT INTO AlimentosTarefa(IDTarefa,IDAlimComEstado)
       VALUES("44","19")

INSERT INTO TarefaOrdenada(IDReceita, numeroOrdem, IDTarefa)
       VALUES("2","3","44")
```

First, it has to create a “Tarefa” (Task) which is composed, in this case, of the action id (“IDAccao”) that is 73 for “Juntar”. The other fields are used in other types of actions, and are filled with “0” when are not used. After this, an association between food and a task is made. In this case there is only one food, so only one insertion is made in the “AlimentosTarefa” table, having “IDTarefa” with value “44” (this is the id from the previous query) and “IDAlimComEstado” with value “19” (that holds “açúcar” with no state associated). Finally, an order (“numeroOrdem” field) is associated with the task (“IDTarefa” field) in table “TarefaOrdenada”, which also links to the recipe (“IDReceita” field).

4 Evaluation

The system was tested in terms of precision and recall with eight recipes (2 of each type, soup, meat, fish and dessert). The results were obtained after the second module (i.e., after obtaining the dependencies) for ingredients and actions parts and after the introduction in the database. To test if an ingredient or an action is correctly extracted, the corresponding dependencies must contain all the information available, which includes all the PARTEDE and ESTADO dependencies, i.e., in the phrase "4 tomates maduros" the ingredient is correctly extracted only if the dependencies INGR(tomates) and ESTADO(maduros) are extracted.

The table 1 shows the number of given, correct and total correct ingredients and actions in the eight recipes, as well as the precision and recall values for each phase.

Evaluation	Ingredients	Actions	Database
Correct Answers	64	87	139
Given Answers	69	110	173
Total Correct Answers	72	135	207
Precision	93%	79%	80%
Recall	88%	64%	67%

Table 1. Evaluation of eight recipes

5 Conclusions

This article shows an information extraction system created for recipes domain, in which the information about ingredients and the actions that constitute a recipe are extracted. The system is rule based, using syntactical analysis to extract dependencies that contains the information of recipes.

With this work it is possible to have information about recipes (ingredients and actions) stored in a database which can be used, for instance, to ask the database about the recipes it has introduced or that information can be used by a dialogue system, like a cooking assistant.

To enhance the results there are some aspects that could be improved, as well as some work that can be done in the future, such as:

- Handling several situations with ingredients like optionality in quantities and ingredients, and ingredients that only have food parts, without any reference to the food;
- Enriching the lexicon of food and utensils, completing the ontological knowledge;

– Creating more rules for actions extraction, considering more situations so that the values for precision and recall could improve;
– Detecting recipe types, such as meat, fish, dessert and soup;

References

1. Ribeiro, R., Batista, F., Pardal, J.P., Mamede, N.J., Pinto, H.S.: Cooking an Ontology. In: The Twelfth International Conference on Artificial Intelligence: Methodology, Systems, Applications. Volume 4183. (September 2006) 213–221
2. Xerox Research Centre Europe: XIP User Guide (2003)
3. Aït-Mokhtar, S., Chanod, J.P., Roux, C.: Robustness Beyond Shallowness: Incremental Dependency Parsing. Nat. Lang. Eng. **8**(3) (2002) 121–144