CeADAR – Centre for Applied Data Analytics Research

Enterprise Ireland Data Analytics Technology Centre

Intelligent Analytic Interfaces: Search & Retrieval - Technical Specification

Document Type:	Technical Spec
Project Title:	CeADAR
DDN Theme:	1 – Intelligent Analytic Interfaces
Theme Leader:	Sarah Jane Delany (DIT)
DDN Sub-Theme:	1.1 – Ease of Interaction
Authors:	Eoghan O'Shea & Robert Ross
Document Version:	1.0
Date of Delivery to ISG:	February 2015
Number of pages:	8

ABSTRACT

Our Search & Retrieval system, SmartSearch, will provide a method for searching knowledge bases that goes beyond simple keyword search techniques. This document will provide a high-level technical specification of the planned system.

CeADAR is a research partnerhip comprising University College Dublin, University College Cork, and Dublin Institute of Technology.

http://www.ceadar.ie

Contents

1 Description of Industry Needs **3**

2 System(s) Involved **3**

3 Approach **5**
3.1 Vector Space Search 5
3.2 Manually-filled Ontologies 5
3.3 Automatically-filled Ontologies 6
3.4 An Intermediate Approach 7

1 Description of Industry Needs

In many companies, the search and retrieval of information matching certain criteria is a key part of the business process. If we take, for example, those companies that work with many different services or products, the task of searching through product or service queries can be a key part of their data analysis and testing processes. Typically in such organisations search is done using manually selected search terms, which can be time consuming and ineffective as a typical keyword-based search will not retrieve the wealth or depth of information required.

From speaking to industry, we have identified two areas of particular interest that would go beyond the limits of simple keyword-based search. The first involves the ability to be able to search for and group items in internal knowledge bases based on a single point of similarity between these items. This point of similarity may not be evident from the names of the items themselves and the challenge lies in drilling down into the item's characteristics, e.g. their composition, in order to find this point of similarity, thus allowing grouping of items.

As an example, in a company that deals with text descriptions of food items, there may be a need to extract food products from a database that are part of the same or similar food group, e.g. all cheeses, all breads, all biscuits, etc. Additionally, there may be a need to extract all food items that contain a particular food component in any form, e.g. as an ingredient, as well as the proportion of the food item that the particular food component comprises. If that component were some form of cheese, say, the search would have to return all the food items that contain cheese as one of its constituent parts, e.g., as a component in pizzas, sauces, cakes, etc., as well as the proportion or fraction of the food made from cheese. We can define this problem as the "source problem".

The second area of interest involves being able to search for and categorise items as belonging to a specific broad area. If we return to our example involving food products, inputting a particular broad food category, e.g. Breakfast Cererals, into search should extract all the food products that would logically be present under that category, so, things like Cornflakes, Rice Krispies, Bran Flakes, etc. We call this the "basketing problem"

This document provides a high-level technical specification of a Search & Retrieval system, which we have named SmartSearch, that seeks to solve the two problems outlined above.

2 System(s) Involved

We can consider a typical use-case of the system to be developed. The use-case actor would be a worker in a food ingredients testing company. The event that would trigger this use-case is that the worker needs to determine the amount and proportion of a particular ingredient in a range of similar food products. If that ingredient is a fruit or fruit extract, then the worker needs to know the foods or recipes that contain this fruit in any form. For this use-case, let us imagine that the worker needs to determine the amount of citrus fruits in different foods. In order to extract this information, the worker makes use of a search and retrieval system that has been configured for the food ingredients domain, e.g. with the knowledge base populated with food information. For the worker to carry out their task, there is a precondition

that the system must have a knowledge base configured so that information on food products can be easily and reliably returned. In addition, it must have a suitably large knowledge base, so allowing the worker to access all the different food products of interest. The search and retrieval system would have a front-end allowing the written entry of search terms. The worker would enable a search by entering the name of the ingredient being searched for into a search box and pressing the on-screen "search" button. The subset of foods found to contain this ingredient would be displayed on-screen, with information on their composition contained in a drop-down box. Where possible information on the percentage/fractional amount of the searched for ingredient in each food would be displayed beside that food's name. It is likely that on occasion certain foods are missing from the knowledge base or that ingredients are not listed or are incomplete. In these cases, information on the foods may need to be added to the knowledge base of the system. The system must be flexible enough so that products that are found not to be present can be added subsequently by a suitably trained staff member. The user inferface would need to be configured to include this option; perhaps allowing the uploading of XML files or similar.

The worker accessing our system may also be interested in knowing the ingredients in the constituent parts of a food, e.g. in a cake the worker may want to separately know the fruit content in the jam filling as well as on the topping or in the layers. The system would have to allow a further drill-down into particular constituent parts of a food product, allowing the proportion of individual food elements in these constituent parts to be listed and returned. The challenge, therefore, is to produce a system that allows the decomposition of foods into their specific ingredients, and moreover, the further decomposition of any of these ingredient into their constituent parts, if required. A further problem to overcome is one of language, specifically the problem of dialect variations or the possible use of synonyms to describe a food, e.g. eggplant for aubergine or pasta for spaghetti, etc.

In the basketing problem, a worker may wish to group food products under a number of broad headers, e.g. all baby food products grouped together, all fried potato products grouped together, etc. In this case, we envisage that a worker would input the broad category of products they wished grouped together, e.g., baby food products. Based on information stored in its knowledge base, where a large range of food products are already listed, the search & retrieval system would find all foods that belong to the relevant group. In the front-end interface, the list of foods found to belong to the broad search term would be printed to the screen.

In this work, we make the assumption that a suitable knowledge base of food information is available. The use of screen-scrapers (or web scrapers) to collect recipe/food information in order to populate such a knowledge base is considered necessary by us. We do not see another way to collect the depth and breadth of knowledge about different food types that a system of this sort requires. Use will also be made of food recipe APIs such as the Yummly recipe API [1] or the Tesco Labs API [2], where available, as well as publicly accessible ontologies. A final commercial version of the system developed here may prefer to use a royalty based API system in order to populate a suitable knowledge base.

[1] `https://developer.yummly.com/`
[2] `http://www.tescolabs.com/?p=7171`

3 Approach

The SmartSearch system will be developed as a search engine contained within a web-based interface. The intention of this system is that a user can input the word they wish searched for in a Google-like front-end, e.g., inputting "apple" for foods containing apple, etc. The system will return a list of food items containing apple in any form among their ingredients as well as the proportion of the apple component in the food (if this information is availabe). To create such a system, we have considered a number of approaches, ranging from a system involving a Vector Space-type search engine to more knowledge-rich systems involving either manually or automatically-filled ontologies, which we will detail below.

3.1 Vector Space Search

A search in vector space would involve converting documents (or, for example, ingredient lists) into vectors. Each dimension within the vectors would represent a different term. For example, these terms could be words describing a recipe, so words like "flour", "sugar", "eggs", etc., and the vector quantities could be a count, say, of the number of these terms in each document. In order to query a document, the cosines between a query vector (containing terms to be searched for) and the document vector can be computed. If a document contains the terms in the query vector then the value of the cosines corresponding to these terms is greater than zero (a precise threshold value for the relevance of a word can be defined). Values where cosines are greater than the threshold are considered to be positive "hits", the search being successful in these instances. Such a system would allow for individual words (terms) in recipes to be searched for. [Note we shall use the word "recipe" in this report to refer to the individual components (ingredients) of a commercial food product as well as its more traditional meaning, i.e., a list of ingredients and cooking instructions for a food that is to be prepared by hand.]

Disadvantages to this approach lie in the fact that this form of simple search would not be able to perfom any form of sophisticated search based on reasoning. For example, while a search for citrus fruit (the search term) would successfully return foods with the word citrus fruit mentioned as part of the ingredient list, it would not return ingredients that contain citrus fruits if the word "citrus fruit" is not specifically part of their name. For example, we might expect a search for foods containing citrus foods to return a product like marmalade (a fruit preserve made with oranges) as one of these. However, although containing a citrus fruit, orange, marmalade is unlikely to include the word "citrus fruits" in its ingredient list. This is a significant shortcoming, particularly in light of the problems we seek to solve. More sophisticated alternatives must therefore be considered.

3.2 Manually-filled Ontologies

A more sophisticated alternative to vector space search would involve the use of ontologies. In an ontology citrus fruits would sit as a subclass of a more general fruits class. Similarly fruits like oranges, lemons, etc. would sit as sub-classes of the citrus fruits class. The benefits of such an ontology are clear. For example, a search in such an ontology for citrus fruits would return not just foods containing that word but all subclasses as well, so

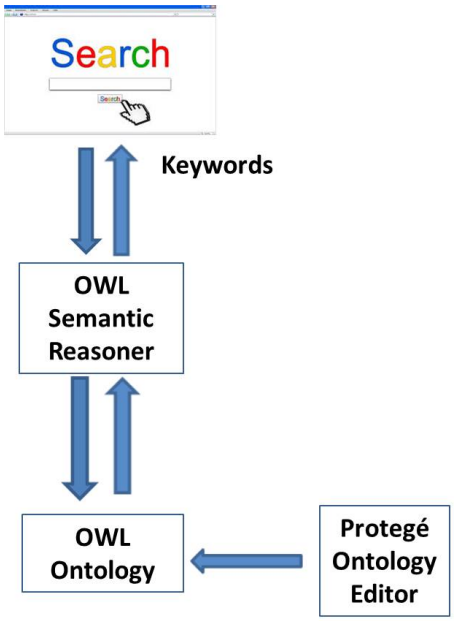

Figure 1: *The key components of the manually filled system.*

oranges, lemons, mandarins, etc. This would overcome the restriction of the vector space search method as in this case all derivations of citrus fruits would be searched for.

Let's consider a system involving manually-filled ontologies. Such a system would require the use of an interface such as the Protegé[3] ontology editor to fill an ontology with a selection of recipes. The created ontology could then be queried using a semantic OWL reasoner such as Pellet[4] called from a suitable front-end interface. The layout of such a system is shown in Fig. 1, whereby a user using a Google-style interface would input the keywords to search for. These keywords would be interpreted by the semantic reasoner and a suitable search term would be sent to the ontology. The results would be returned from the ontology to the reasoner and from there to the user using the front-end interface. The drawbacks of such a system lies in the fact that to expand its coverage (the number of recipes) would require a user to have knowledge of Protegé and the time required to manually input each recipe as required. While this system would work, it would not be, in our opinion, an ideal solution as it would be limited to a small number of recipes, at least initially, and it would be difficult to expand.

3.3 Automatically-filled Ontologies

A more complete solution would involve Natural Language Processing (NLP) based ontology creation, e.g., as in this paper[5]. Using such systems would allow us to automatically create food ontologies based on information on recipes scraped from different websites or accessed from various APIs. Creating such a system and linking it to a Pellet Reasoner and an appropriate front-end would allow us to solve the food source and basketing problems.

[3] `http://protege.stanford.edu/`

[4] `http://clarkparsia.com/pellet/`

[5] Maynard, Diana, Adam Funk, and Wim Peters. "SPRAT: a tool for automatic semantic pattern-based ontology population." International conference for digital libraries and the semantic web, Trento, Italy. 2009

However, while this would be an ideal solution, it is our opinion that the development of such a system lies outside the 6 month window allocated to this project, requiring a significantly longer development time.

3.4 An Intermediate Approach

The system we will develop is an intermediate one between the last two ontology options outlined above. A process of scraping or information retrieval from websites and APIs will be carried out to first create a knowledge base of each of the food items in a pre-agreed list. We propose to use Scrapy[6] for this task, the output of which should be a database with an XML data structure. Having the data in a form such as XML allows the database (knowledge base) to be searched and different data mining techniques applied, e.g. to extract a particular ingredient from a recipe. Together with this XML-based knowledge base, we will re-use presently available food ontologies available on the web, such as that shown in this footnote[7]. Through a combination of the XML database and the available online ontologies, different food ingredients can be searched for and associated to various food products, answering the problems set out above.

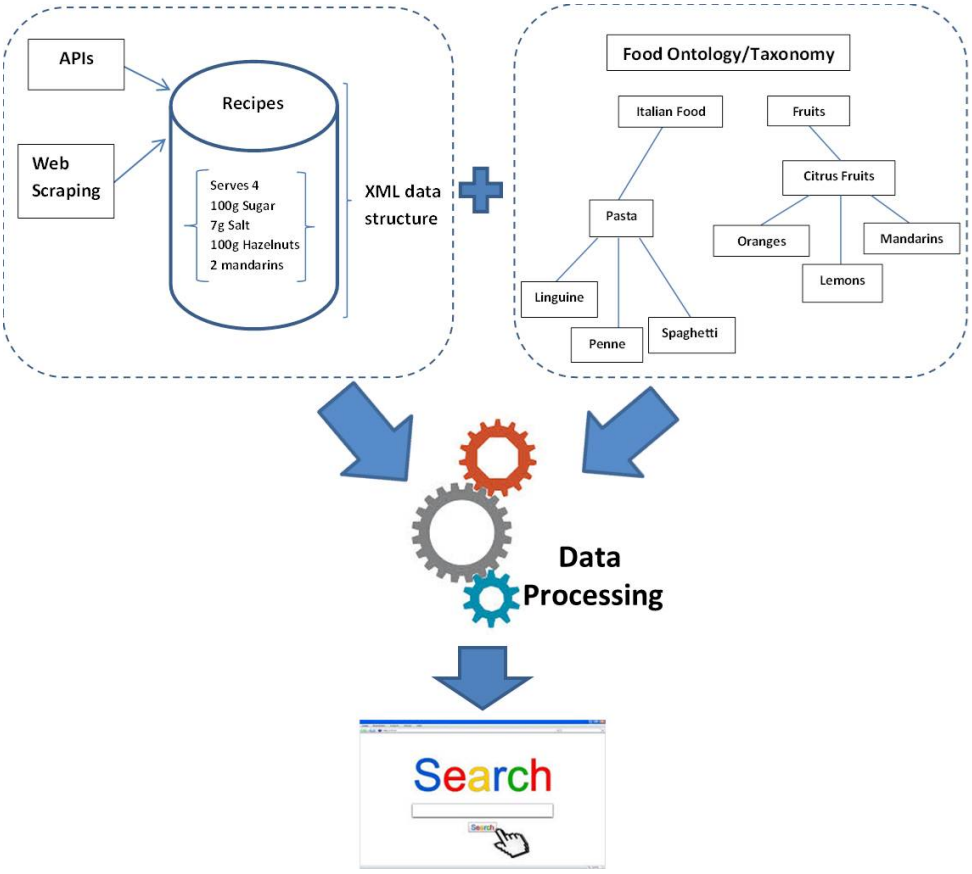

Figure 2: *The components of the proposed "intermediate approach" ontology based system.*

Fig. 2 shows a general overview of the system. Based on this figure, and assuming we were interested in finding all products containing citrus fruits, we could search within the avail-

[6] `http://scrapy.org/`

[7] `https://raw.githubusercontent.com/ailabitmo/food-ontology/master/food.owl`

able ontology/taxonomy and extract the subclasses of citrus fruits from this ontology. This would return things like mandarins, oranges and lemons. Based on these extracted words, this would allow us to search all recipes stored in our knowledge base that contain one or more of these words. This would solve the food source problem. Similar techniques, involving data mining of the XML database, in conjunction with the use of available ontologies, should allow us to solve the basketing problem.