Ontology Based User Interface

In general [15], developing user interfaces hindered by the knowledge of the user. Letting the user known what can he ask for and constructing a meaningful search query using the user interface is the major issue. To remedy this issue, solutions have been proposed. Some of those solutions would be making every option in the user interface available to reduce the recall issue. Another solution would be writing manuals to the user to follow. These solutions might be providing more complexity and other problems. The former solution could overwhelm the user with all of the options available whether needed options or not. The latter solution could increase the load on the user to study and spend time on something that needed to be recalled eventually. Ontology driven user interface would be the most suited solution, since ontologies are based on conceptual model rather than just terms [3, 15]. This conceptual model gives a map for the user to follow upon constructing queries [15].

Ontology based user interface is defined by [3, 15], as a user interface that allows the user to construct and manipulate queries based on some domain concept stored in ontology. This domain concept drives the user interface, where there is no need for manuals or shove all available options in the user interface, since the ontology based one which should act as a guide for the user. It depends on recognizing knowledge instead of memorizing keywords. It allows the user to build complex and meaningful queries and return the needed results. In addition, it offers the user the option of browsing around to find out what he/she needs. The user does not have to any thing about the underlying conceptual knowledge. TAMBS give the illusion of retrieving from single source while it read from multiple sources and convert selected options to appropriate query languages that match sources'.

Keywords are stored in hierarchical taxonomy so user choose rather than recall. In ontology based UI the keywords are to be called categories, and each category may have subcategories. To make it more formal, let us assume C_i is a selected category, where (C_i = 1, .., n) representing all categories selected, and C_i is also represents $(S_{i,1}, \dots, S_{i,k})$. Each category may consist of subcategories. The search query in DL format might be represented in conjunctive normal form (CNF):

$$(S_{1,1} \vee \dots \vee S_{1,k}) \wedge (S_{2,1} \vee \dots \vee S_{2,k}) \wedge \dots \wedge (S_{n,1} \vee \dots \vee S_{n,k})$$

For example, in the pizza finder application user may choose VegetableTopping to get all pizzas that are have vegetable on them. In addition, BaseClass, which is NamedPizza, is included by default. As a result, we have two categories NamedPizza class and the equivalent class of any pizzas that have VegetableTopping. NamedPizza content subcategories like Four Seasons, American, and etc. Where the equivalent class of any pizzas, which have VegetableTopping, could have something like Four Seasons, Soho, and etc. Basically, taking the result of their CNF will be the result. The result will include Four Seasons since both of the categories intersect there, and may intersect in other pizza.

The user interface offers choices and some scenarios for the user, so that the user would be guided toward constructing meaningful queries that return the intended results [3, 15]. Users would not face the no-result status after running queries. Query expressions are Description Logics (DLs) expressions and they are incremental and compositional [3, 15]. DL is a way for knowledge representation used by the conceptual model [16]. It provides hierarchal model based on conceptual model that represent classes of specific domain and the relationships between the instances of those classes [3, 15]. DL model is not easy because of the need knowledge about the DL syntax along with understanding it, so a friendly user interface need to build to separate the user from dealing with DL [15].

According to [15], there are two kind of concepts that the DL model support. The concepts definitions and the assertions made on the concepts definitions, like the subsumption relationship between two classes. In a way, assertions on the original concepts considered as defining new concepts definitions. Compositional concept can be formed using some services provided by DL. Reasoning about the concept definitions is done through the services provided by DL. These services are:

- **Satisfiability**: make sure that the concepts are consistent.
- **Subsumption**: create composite concepts definition from assertion made on the original concepts definition.
- **Classification**: make new classification hierarchy based on the subsumption relationship.

- **Retrieval**: retrieve any individual that is part of concept definition whether it is original or generated from the subsumption relationships.

Ontologies support creating annotation properties within themselves and associate them with entities [17]. These annotations properties consist of the name and the value. Annotations within ontologies play a major role in driving the user interface. Annotations would form some set of rules for the user interface to follow and interact based on. A tool was developed based on some animals ontology and annotations would be a good example of what could annotations do in term of user interface interactivity. This tool allow to brows for Apes' photos based on some annotated criteria such as the quality of the image and the environment where it been taken.

In this project, I am trying to develop an application to find out different information about sushi. The application is ontology based. Annotations play major role in driving the user interface to make more flexible. This application is based on The Manchester Pizza Finder application, which will be elaborated on a separate section, except this one would have some enhancements.