

A Framework for Extending Fuzzy Description Logic to Ontology based Document Processing

Shailendra Singh Lipika Dey Muhammad Abulaish

Department of Mathematics
Indian Institute of Technology, Delhi,
Hauz Khas, New Delhi, India
Email: (lipika, shailen)@maths.iitd.ernet.in

Abstract. Ontologies have proved to be very useful in sharing concepts across applications in an unambiguous way. However, since web documents are not fully structured sources of information, it is not possible to utilize the benefits of a domain ontology straight away to extract information from such a document. In this paper we have proposed an uncertainty handling mechanism based on rough-fuzzy reasoning principles which can handle query-processing from unstructured web documents in an elegant way.

1 Introduction

In order to maximize the utility of web resources, it is essential to extract information from heterogeneous, unstructured web documents and reason with them. Several question-answering systems try to do this, though most of them still just list the relevant URLs. AnswerBus [8] is an open-domain question answering system based on sentence level Web information retrieval. This system formulates a simple query from each user question by deleting functional words, using word frequency table, deleting special words and modifying word forms, if necessary. It then employs five search engines to retrieve relevant texts and then extracts relevant portions from these, reorders them and presents them as answers.

Though these systems work quite well, certain problems related to the nuances of Natural Language remain unsolved. For example we posted two queries to AnswerBus: (i) *What is a wine with fruity flavour, light hop, strong taste?* The answer was *“Sorry, I found no answer for your question”*. However when we asked (ii) *“What is a red wine with fruity flavour, light hop, strong taste?”* there were 3 answers, starting with *“The nose on this beer is ripe and fruity with slightly vinous esters, married to interesting hop notes - the colour of a glorious red ruby wine”*. Clearly, this could be one of the answers for the earlier question also, since *“red wine is a special class of wine”*. We feel that such systems can be greatly benefitted by integrating uncertainty handling mechanisms and underlying ontological structures with them.

Ontology based knowledge representation scheme provides a platform for sharing concepts across various applications in an unambiguous way [1], [2], [3], [4], [5]. Description logics have proved to be a useful tool in ontological engineering. However, in order to reason with contents of web resources, it is essential to extend Description Logic to query answering services [9], [10]. Since natural language resources contain imprecise, often vague or uncertain descriptions of concepts, for practical applications to question answering it is essential to extend Description Logic to handle imprecision. As another example, we consider the another question “*What is a red wine with fruity flavour, light hop, not light taste*”? One of the answers was “*The mystical magic of this well is reflected in the rich golden colour, fruity flavour and full hop aroma of this light and refreshing ale.*” Clearly this is not a desirable answer. The problem lies in the interpretation of terms “not light” against “light” or “strong”. Since the answers are based on string matching the fact that “not light” is closer to “strong” than “light”, cannot be taken care of.

Looking at these problems of dealing with natural language texts, we were motivated to design a document analysis system which can do the following

- Identify implicit concepts in a document using underlying Ontological structures and treat words as concepts rather than patterns
- Handle imprecision of natural language texts arising due to linguistic qualifiers through the use of fuzzy Description Logic [11], [12]

[9] has shown how by placing certain restrictions on the use of variables in a query, Description Logic can be extended to DAML+OIL. They have focussed on Boolean queries to determine whether a query without free variables is true with respect to a Knowledge Base. Holldobler et al. [11] proposes a fuzzy Description Logic with Hedges as concept modifiers.

In this paper, we have proposed a framework for reasoning with imprecise and unstructured knowledge sources with an underlying ontological structure. The system reasons about relevance of documents using a rough-fuzzy reasoning mechanism.

Though a lot of focus is currently given on standardizing ontology representation languages for general domain representations, most of these assume that the concept world is exactly defined in terms of properties and values. The problem with this approach is that such a structure cannot be used straight away for retrieving information from an unstructured text document.

The salient features of this work are:

- We have provided a complete framework for incorporating fuzzy concept modifiers into an ontological structure.
- We propose a reasoning mechanism which can be used to match document concepts with user given concepts more satisfactorily by increasing the scope of the question to include related concepts.
- Finally a relevance computation function can be used to compute the overall relevance of a document using the extended query.

2 Enriching Ontological Description with Linguistic Qualifiers

Ontology provides a shared conceptualization of domains to heterogeneous applications. Usually a concept is defined in terms of its mandatory and optional properties along with the value restrictions on those properties. Figure 1 shows the elements of an ontological framework and Description Logic.

We find that, in general there is no ontological framework for qualifying a property. Even if linguistic variables like low, high etc. are used, they are most often used as value restrictions. For example, the wine ontology [3] defined by W3C group, uses the *properties hasflavour, hascolor, hasbody* etc. for describing any class of wine. The value restrictions on *hasflavour* is the set {Delicate, Moderate, Strong} and that of *hascolor* is {Red, White, Rose}. This ontology also describes a set of instances of wines of various types and describes them in terms of the above mentioned properties. Using Description Logic it is possible to retrieve information from this knowledge base.

However, some web documents describing wine are produced below:

- “BARBERA is a full bodied, fairly acidic red wine grown originally in California's coastal areas is now the major component of the red jug wines produced in California's Central Valley”;
- “CABERNET FRANC is a light, fruity red wine from France's Loire Valley, often blended with Merlot. Basically used as a blending grape in California, especially with Cabernet Sauvignon.”

If the user is looking for a wine that is “medium bodied, not acidic, light flavoured”, then definitely CABERNET FRANC is a better choice than BARBERA.

Considering these we propose a framework in which a general set of concept modifier can be associated to ontology. The role of these modifiers will be to change the degree of a property value. This set is usually a partially ordered set [11]. For example, the hedges “pale, slight, light, more or less, very, deep, dark, rich” can be associated to the wine ontology. We change the wine description ontology to include in addition to its original definition, the description of linguistic qualifiers and their associations to the property values. We accomplish this through multiple inheritance of each “Winedescriptor” property. “FuzzyQualifier” is a new subclass which can take values from the set of hedges only. The hedges are ordered by increasing instance number of nodes. Each property like “WineTaste”, “WineFlavour” etc. are subclasses of “WineDescriptor”. Now we introduce fuzzy property classes like “FuzzyTaste”, “FuzzyBody” etc. as subclasses which multiply inherit from “FuzzyQualifier” and the respective class descriptor. This property inherits the possible values from the property descriptor and the concept modifiers from the fuzzy qualifier class. Thus a description of a wine can be generated in terms of a list of concept modifiers followed by a property value. Parts of the OWL code that is generated thereby is shown below:

```
<owl:Class rdf:ID="FuzzyQualifier">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#has-qualifier"/>
```

```

    </owl:onProperty>
    <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:cardinality>
  </owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="FuzzyBody">
  <rdfs:subClassOf rdf:resource="#WineBody"/>
  <rdfs:subClassOf rdf:resource="#FuzzyQualifier"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#has-qualifier"/>
      </owl:onProperty>
      <owl:maxCardinality
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >8</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#has-value"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#WineBody"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:about="#has-qualifier"/>
      </owl:onProperty>
      <owl:allValuesFrom rdf:resource="#FuzzyQualifier"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  .....
<j.0:FuzzyQualifier rdf:ID="Instance_56"
  j.0:has-qualifier="not"/>
<j.0:FuzzyQualifier rdf:ID="Instance_57"
  j.0:has-qualifier="mild"/>
<j.0:FuzzyQualifier rdf:ID="Instance_58"
  j.0:has-qualifier="moderate"/>
<j.0:FuzzyQualifier rdf:ID="Instance_59"
  j.0:has-qualifier="very"/>

```

Fig. 1. Elements of an ontological framework and Description Logic

With this definition an entity can have property value \mathcal{CA} , where \mathcal{C} is a list of concept modifiers. This can be interpreted as having the value A to a certain degree. For example the wine BARBERA which has taste “fairly Acidic”, has

certain degree of membership to the “Acidic Taste” category which is definitely different from 1. With a predefined set of concept modifiers, given an entity description in terms of these, the relevance of the entity to a user query can be computed as a function of its membership to the user given description. Thus if a user asks for a “very strong” wine, the relevance of two wine descriptions which are “light” and “very, very, strong” respectively, can be computed.

2.1 Computing Relevance of Property Values with Concept Modifiers

Let $H = \{h_1, h_2, \dots, h_p\}$ be a set of hedges, among which each element of H is either positive or negative with respect to all other hedges including itself. The function $\text{sign}: H \times (H \cup \emptyset) \rightarrow \{-1, +1\}$ is defined in [11] as

$$\text{sign}(h_i, h^*) = \begin{cases} -1 & \text{if } h_i \text{ is negative wrt } h^* \\ 1 & \text{if } h_i \text{ is positive wrt } h^* \end{cases} \quad (1)$$

The set of fuzzy qualifiers operates with hedge algebra as defined by [11]. An entity can have property value \mathcal{CA} , where \mathcal{C} is a list of concept modifiers. $\mathcal{C} = h^*$, where $h \in H$. This membership value of \mathcal{CA} can be calculated using the algorithm presented in [11]. Using this algorithm one can compute the membership of any arbitrary combination of the concept modifiers.

Table 1. Membership calculation for concept modifiers

	<i>Sig</i>	β
<i>Strong</i>	1	2
<i>Light</i>	-1	0.5
<i>Very</i>	1	2
<i>Mol</i>	-1	0.5
<i>Very Strong</i>	+1	4
<i>Very light</i>	-1	0.25
<i>Mol Light</i>	+1	0.75

In our adaptation of this approach, we have made the following modifications. The set H is defined as follows:

$H = \{h_0, h_1, \dots, h_p\}$ where $h_0 = \text{not}$. While the remaining set of modifiers behave in the same way as before: the sign of h_0 is defined as follows:

$$\text{sign}(h_0, h^*) = \begin{cases} -1 & \text{if } \text{sign}(h^*, A) = -1 \text{ for any } A \\ 1 & \text{if } \text{sign}(h^*, A) = +1 \text{ for any } A \end{cases} \quad (2)$$

Table 1 below shows the computation of a four member set containing {strong, light, very, mol} with partial relationships defined between strong and light and between very and mol, where mol stands for “more or less”. Thus if it is given that “a wine is red has value 0.7”, then the membership of “the wine is very red” = 0.49, “the wine is not red = -0.7”, “the wine is not very red = -0.49”. This rightly interprets than it is more correct to say that the wine is not very red than to say the wine is not red.

3 Accepting User Query

The query is assumed to have a mandatory component – entity name, followed by an optional list of property specification. The property specification is a variable length list of property qualifiers and values. Thus a query assumes the following structure: Query: [Entity name: [{Property}: <List of Qualifiers>* <Property value>]*]. Thus a sample query in the wine domain can be

```
[wine:[{Flavour}:<Light>(Fruity)
      [{Colour}:<very deep>(red)]
      [{Taste}:<Very strong>(acidic)].
```

For queries pertaining to an arbitrary entity that does not belong to any defined ontology, currently the system accepts any string given as property name and value. However, the concept modifiers can function correctly only if they are part of the defined modifier resource. For example looking for a bright celestial body resembling a comet, without any underlying ontology one can construct the query as:

```
[celestial body: [{appearance}<bright>
                  {like}<very much><comet>].
```

Since our emphasis is on ontology based query processing, the system expects concepts to be from a particular domain. If so, then it is logical to expect that the entities are described in terms of a pre-defined set of concepts. Our query-processing framework exploits this structure along with the query approximations constructed above, to locate property values.

4 Query Extension using Related Concepts

Question answering systems usually extend a given user question by extending the user query to include relevant words. The set of words to be included for a query is usually a predetermined set derived from works on Question classification[13]. For

example, [13] mentions 15 categories of questions, how they can be identified and extended.

In our work however, we extend the query with ontologically related concepts. Since ontologies are generated by domain experts, this leads to a better approximation of query. The ontologically related concepts to be included into a query is determined from the type of the question.

Along the lines of rough approximation, we define two approximations of the question – the lower approximation and the upper approximation. Let Q be the user query consisting of a set of concepts C .

Definition 1. Lower approximation of Q includes all those concepts which are semantically equivalent to the concepts. Mathematically,

$$\underline{Q}(C) = \left\{ c' \mid c' \equiv c \text{ where } c \in C \right\} \quad (3)$$

If the underlying ontology is the English language, then all words synonymous to the original set of words will be included in $\underline{Q}(C)$.

Upper approximation of Q can include all concepts which are related to the concepts in the original query through the class-subclass relations. We define two types of upper approximations that can be obtained depending on the type of the query.

Definition 2. The generalized upper approximation of a query Q , denoted by $\overline{Q}_G(C)$, extends the set of query concepts C by including all concepts which are super classes of the original concepts. Thus

$$\overline{Q}_G(C) = \left\{ c' \mid c \subseteq c', \text{ where } c \in C \right\} \quad (4)$$

Definition 3. The specialized upper approximation of a query Q , denoted by $\overline{Q}_G(C)$, extends the set of query concepts C by including all concepts which are super classes of the original concepts. Thus

$$\overline{Q}_G(C) = \left\{ c' \mid c' \subseteq c, \text{ where } c, c' \in C \right\} \quad (5)$$

The query type will determine which type of approximation is to be taken. For example, let us look at the following queries:

1. Get a wine with a fruity flavor: If fruity is replaced by anything that is a fruit like orange, strawberry, grapes etc. the quality of answer remains same.
2. What is a fruit? : If fruit is replaced by a subclass like citrus fruit or berry, the answers retrieved with the specialized concepts may not give the exact answer.

We are yet to find a satisfactory way of determining which approximation is better under given circumstances. We work with the union of both the approximations. Since concepts in the original query and those included in the approximated extension cannot have the same weight while calculating document relevance, we associate weights to the concepts in the approximated query.

Let the original query be denoted by \mathbf{Q} and let \mathbf{A} denote the approximated query. The weight of any concept $\mathbf{c} \in \mathbf{A}$ is computed in terms of its closeness to concepts in \mathbf{Q} .

Thus $\dots \mathbf{c} \in \mathbf{A}$, $w(\mathbf{c}) = \sup_{\mathbf{d} \in \mathbf{Q}} \{1/(\text{dist}(\mathbf{c}, \mathbf{d}) + 1)\}$, where $\text{dist}(\mathbf{c}, \mathbf{d})$ is the number of classes lying between \mathbf{c} and \mathbf{d} in the ontology tree, along a class-subclass hierarchy.

The above measure ensures that if \mathbf{c} is a concept from the original query it is retained with weight 1.0. New concepts are inducted with a weight that is determined by the minimum distance of the concept from one of the query concepts.

5 Query Processing Framework

To find the relevant documents, we compute the similarity between the extended query and the unstructured texts. Since we have not implemented a query classifier, presently the user query is assumed to be in a structured format.

To extend a query with its approximation, we consider all values in the original query, and extend it to include other terms from the ontology. Thus we include all classes which are either generalization or specialization of the original value concepts, in conjunction to the appropriate property.

Locating Relevant Concepts in Document. Each document is considered in its text form as a collection of words. HTML tags are deleted in case it is an HTML document.

Since a document as a whole may not be relevant to the query, we first try to locate appropriate portions in the document which are likely to be relevant to the query. We call the relevant portion the “*property window*”. For each property specified in the query the property window for that property is the portion of the document which contains information about it. In our implementation we have used a sentence containing property names as a property window. A single property window may be verified by multiple property queries since a single sentence can have more than one property definition in it. Once we find the property windows, we look for qualifier-value matches within the window. Those properties for which a window could not be located, we consider the values in the approximations of the query and try to locate them in the document. These values are considered to be relevant only in conjunction with the properties that have not been already located. The sentences which contain the values serve as “*value windows*” for these properties. Once again, the same value window may be chosen for multiple properties since in absence of a specific property name, it is not possible to decide the value is pertaining to which property.

Computing relevance in terms of value matches between query and document:

In this phase we initiate an activation mechanism to locate the query values within property windows. A value may be a concept in the original query or in the extended query. When a value concept is located in the document, a search for the modifiers preceding it is initiated. This may yield a list of modifier concepts. The final match is expressed as a similarity measure between the original query concepts and the concepts found in the document.

Let δ be the concept modifier for c in original query. Let β be the power determined for δ using algorithm of [11]. We find the concept modifier in the document for the related concept by initializing η to blank and then while earlier token in the sentence is a hedge $h \in H$, we concatenate h to η so that η becomes $h\eta$. We then determine β' , the required power for η . The relevance of the value is then computed as

$$\text{Rel}(n) = 1, \text{ if } w(n)=1 \text{ and } \beta=\beta' \\ \text{/* denotes an exact match for value as well as qualifier*/} \\ \text{else } \text{Rel}(n) = w(n).e^{-|\beta-\beta'|}$$

This value is always less than or equal to 1.0, for each concept n . The final relevance of a document is computed as a normalized value of all matches with concepts in approximate query.

For example, let us consider the query [Wine:<requirements for flavor are - <light fruit>+<strong malt> + <mild citrus>]. We show the relevance with respect to single sentences picked up from documents on beer from www.RateBeer.com.

- Beer1 - A wine that has aroma of *moderate malt*, *light fruit*, with notes of orange peel, toffee, light note of alcohol - relevance **1.0**
- Beer2 - This wine has aroma is which is of *strong fruit*, with notes of apple, toffee, light note of alcohol – relevance = Relevance = **0.078**.

We consider the query stated earlier rephrased as [Wine:[<Flavour><Fruity>[[<colour><very><red>][<hop><not><full>]]. Consider the documents

- Document 1 - These wines can be from very pale pink to a *light red* in colour, fresh and *fruity* in their flavour, balanced with acidity and the alcohol should be from 9% to 12% by volume. **Relevance = 0.539**
- Document 2 - The mystical magic of this well is reflected in the rich golden colour, *fruity* flavour and *full* hop aroma of this light and refreshing ale. **Relevance = 0.524**.

We can see that the concept modifiers can be taken care of in this framework and thus the original ordering can change to give better matches.

However, the key problem that remains is the virtually endless number of concept modifiers and property-value pairs that can crop up in a domain, which cannot be taken care of in an ontology. We are therefore working on a system which is initially trained with a large number of documents, where the user identifies the values and qualifiers. Using this initial set, we are building a database of properties and their possible values. We have integrated this system to a tagger which can identify the nouns, adjectives and adverbs. This is used to reduce the set of words to be searched for. Finally we are also exploring the possibilities of utilizing a Hidden Markov Model for analyzing the documents to give a better precision.

6 Conclusion

In this paper, we have presented a basic framework for integrating concept modifiers in an ontological framework. The ontological structures are enhanced with concept modifiers to model linguistic qualifiers since we found that documents rarely contained isolated values. We have also shown how query approximations can be done by extending the original query to include additional concepts from the ontology. This can enable extraction of more relevant documents through the search engines by using relevant concepts also. Finally, we have shown how relevance of documents can be computed using concept modifiers. We have provided an initial framework for query processing and providing to the user the degree of relevance of a document. However, a lot more needs to be done to accommodate unmatched concepts and we are working on integrating a learning mechanism to improve the accuracy of the system.

References

1. Zhong, N., Liu, J., Yao, Y. Y.: In search of the wisdom web. *IEEE Computer*, Vol. 35(11). (2002) 27-31
2. World wide web consortium. <http://www.w3.org/>
3. WineOntology.
http://kaon.semanticweb.org/Members/rvo/WebOnt_Guide/wines.owl/view
4. Gal, A., Modica, G., Jamil, H.: Improving web search with automatic ontology matching. <http://citeseer.nj.nec.com/558376.html>
5. Ramsdell, J. D.: A foundation for a semantic web.
<http://www.ccs.neu.edu/home/ramsdell/papers/swfol/swfol.pdf>. (2001)
6. Srinivasan, P., Ruiz, M.E., Kraft, D.H., Chen, J.: Vocabulary Mining for Information Retrieval: Rough Sets and Fuzzy Sets. *Information Processing and Management*, Vol. 37. (2001) 15-38
7. Ross, T.: *Fuzzy Logic for Engineering Applications*. McGraw-Hill Book Company, New York, NY, ISBN: 0-07-053917-0. (1995)
8. Zheng, Z.: AnswerBus Question Answering System. *Human Language Technology Conference (HLT 2002)*. San Diego, CA (2002)
9. Horrocks, I., Tessaris, S.: Querying the semantic web: a formal approach. In *Proceedings of the 13th International Semantic Web Conference (ISWC 2002)*. *Lecture Notes in Computer Science*, Vol.2342. Springer-Verlag (2002) 177-191
10. Horrocks, I., Tessaris, S.: A conjunctive query language for description logic aboxes. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI 2000)*. (2000) 399-404
11. Hölldobler, S., Khang, T., D., Störr, H., P.: A fuzzy description logic with hedges as concept modifiers. In *Proceedings InTech/VJFuzzy'2002*. Science and Technics Publishing House, Hanoi, Vietnam (2002) 25-34
12. Zadeh, L., A.: A fuzzy-set-theoretic interpretation of linguistic hedges. *Journal of Cybernetic*, Vol. 2. (1972)

13.Li, X., Roth, D.: Learning Question Classifiers. In Proceedings of the 19th International Conference on Computational Linguistics. (2002)