

Information Extraction and Imprecise Query Answering from Web Documents

Muhammad Abulaish^a and Lipika Dey^b

^a*Department of Mathematics, Jamia Millia Islamia (A Central University), Jamia Nagar, New Delhi-25, India*

E-mail: abulaish@computer.org

^b*Department of Mathematics, Indian Institute of Technology Delhi, Hauz Khas, New Delhi-16, India*

E-mail: Lipika@maths.iitd.ernet.in

Abstract. Word based searches for relevant information from texts retrieve a huge collection and burden the user with information overload. Ontology based text information retrieval can perform concept-based search and extract only relevant portions of text containing concepts that are present in the query or those that are semantically linked to query concepts. While these systems have better precision of retrieval than general-purpose search engines, problems arise with those domains where ontological concepts cannot be unambiguously described using precise property descriptors. Besides, the ontological descriptors may not exactly match text descriptions or the user given descriptors in query. In such situations, uncertainty based reasoning principles can be applied to find approximate matches to user queries. In this paper we have presented a framework to enhance traditional ontological structures with fuzzy descriptors. The fuzzy ontology structure has been used to locate and extract both precise and imprecise descriptions of concepts from Web documents and then store them in a structured knowledge base. The design of the structured knowledge base, which in our case is a database, is also derived from the underlying fuzzy ontology representing the domain. User queries are processed in two stages. In the first stage, precise SQL queries are formulated and processed over the knowledge base to find a possible answer set. In the second stage, fuzzy reasoning is applied to compute the relevance of the answers in the answer set with respect to the query. We have provided experimental validation of the approach through knowledge-extraction and query processing executed over a diverse set of domains.

Keywords: Information extraction, ontology, fuzzy ontology structure, imprecise concept descriptor, imprecise query processing.

1. Introduction

The World Wide Web (WWW) is a large and growing collection of texts and the wealth of information available on it is a valuable resource to the Internet users. However, text mining or locating, extracting and analyzing required information from this vast unstructured collection is a challenging task. Most of the problems are obviously due to the incapability of the computer in comprehending natural language texts with all its nuances. Information extraction from texts chiefly employs pattern matching to identify predefined sequences in text. There are several text-mining tools, which can successfully categorize and index a large collection of documents on a given set of keywords automatically. However, pattern matching works only in a very limited context. A more intelligent way to analyze contents of text is to employ concept linkage. Concept linkage aims at identifying commonly shared concepts across documents and helps users to find information that they perhaps wouldn't have found using traditional searching methods. Concept linkage has played a crucial role in mining information from biomedical

texts, where the text information grows at such a tremendous rate that without automated tools for analyzing the collection, it becomes impossible to assimilate the knowledge embedded in it.

Concepts themselves however may be defined in different ways, by different authors. Hence one way of accessing information stored within unstructured or semi-structured text documents is to go for effective semantic analysis of texts with the help of a structured collection of domain concepts in the background. The Semantic Web, introduced in [28] also aims at providing a new direction towards manipulating the meaning of Web data along with the use of ontologies. Ontologies are knowledge-management structures in which key concepts and their inter-relationships are stored to provide a shared and common understanding of a domain across applications [10]. Ontology based text information processing has been probed quite successfully for analyzing biomedical texts, with the help of several biological ontologies like

GENIA¹ ontology, Gene² Ontology (GO), Open Biological³ Ontology (OBO), TAMBIS⁴ Ontology (TaO) [19] etc.

Even as the use of ontology for domain-specific applications are fast gaining popularity, researchers are actively engaged in tackling some of the chief bottlenecks that still hinder the use of ontology for general-purpose applications. Some of these are identified as follows:

- (i) *Reliable Ontologies do not exist for all domains:* Since ontologies are meant to provide shared conceptualization of a domain, these have to be built by domain experts. Hence ontology building is an expensive task. Integrating knowledge acquired from text documents may provide an effective solution to this problem.
- (ii) *Finding the ideal concept description:* It is impossible to describe all concepts unambiguously. For example, while it is possible to agree on a technical definition of proton, it is impossible even for domain experts to agree on what should be the ideal set of values for describing wine colors. The commonly occurring values are red, white and rose. But one does find descriptors like straw, pink etc. used for describing wines. However, absence of an exhaustive set of values does not undermine the necessity for a wine ontology, which provides a unified framework for describing concepts related to wine. What is ideally required is that within the rigid structure of the ontology, which is dictated by the application, there should be the flexibility to adapt new or modified descriptors as novel use of concepts are encountered.
- (iii) *Matching ontology-specified descriptions to documents or user-given descriptions:* Since it is difficult to agree on a unanimous description of a concept, it is unreal to expect that concepts will appear in a document in exactly the same way as they are described in the ontology. This may also be forbidden by natural language construct requirements in many cases. Thus, keyword based searching have to be enhanced with additional reasoning capability to perform concept-based search. Similarly, concept descriptions given in user queries are not likely to be restricted to ontology descriptions only. Thus, if an ontology is to be employed for searching related concepts in texts in response to a user query, uncertainty based

reasoning principles have to be employed for establishing the multi-way relationships among all these descriptions.

It is possible to reason about ontology entities using inexact reasoning principles, without affecting the rigidity of the underlying structure. We illustrate our point using weather descriptions of various countries as depicted in a web document. Following are some sample descriptions of the weather of various countries:

- Belarus: cold winters, cool and moist summers; transitional between continental and maritime.
- Belgium: temperate; mild winters, cool summers; rainy, humid, cloudy.
- Croatia: Mediterranean and continental; continental climate predominant with hot summers and cold winters; mild winters, dry summers along coast
- France: generally cool winters and mild summers
- Greenland: arctic to subarctic; cool *summers*, cold winters

It is easy for human beings to infer that *Belarus* and *Greenland* have *similar climate* while *France* is *somewhat similar* to *Belgium*, and different from the earlier two. However, given the current status of text information retrieval, it is not possible to automatically derive these conclusions through text analysis. It is in fact impossible to do so using simple word-based search mechanisms. Our aim is to provide a platform where such reasoning can be performed. Fuzzy reasoning can help in establishing the degree of similarity between various linguistic qualifiers like *mild*, *cool*, *cold* etc. For example, using such a system, user may get to know *which country has winter similar to that of France*. This requires not only identification of concepts and their descriptions from multiple sources, but also to compare them using uncertainty based reasoning principles. We have found that such descriptions occur frequently in various domains, and hence the methodology can be applied successfully to build various domain-based applications.

In this paper we have proposed the design of a complete knowledge-management system for handling text documents from a known domain, which addresses some of the concerns expressed above. Starting with a base ontology, the system extracts key information through syntactic analysis of text documents in an ontology-guided way. The information components extracted are stored in a structured format to help in future query processing. The system is also equipped with an ontology enhancing mechanism, which enriches existing concept descriptions with new descriptors extracted from web documents. The novelty of the system

¹ <http://www-tsujii.is.s.u-tokyo.ac.jp/~genia/topics/Corpus/genia-ontology.html>

² <http://www.geneontology.org/>

³ <http://obo.sourceforge.net/cgi-bin/table.cgi>

⁴ <http://protege.stanford.edu/plugins/owl/owl-library/tambis-full.owl>

lies in using a new fuzzy ontology structure, which is created as an extension to the standard ontology structure. Traditionally concepts are described in an ontology using a <property_name, property_value> framework. The proposed fuzzy ontology structure stores concept descriptions in a < property_name, property_value, qualifier> framework. This framework allows defining a property_value of a concept with differing degrees of fuzziness, without actually changing the concept description paradigm. We have termed such concept descriptions as fuzzy concept descriptions.

The proposed fuzzy ontology structure is very different from other similar structures [5,15,29] which usually store the strength of a relation as a function of the co-occurrence of two participating concepts. In our case, qualifiers are chiefly fuzzy linguistic variables, which as a special case could be numerical values also. Hence in our framework concepts can be described with varying degrees of fuzziness. An initial version of the proposed framework for storing fuzzy concept descriptions was presented in [13]. However, in [13] ontology enhancement mechanisms were not proposed. The set of values and qualifiers used for describing ontology concepts was fixed a priori. Consequently, only these descriptors were extracted from text. Presently, our system starts with an initial ontology and is capable of enhancing it with more descriptor values and qualifiers. Knowledge enhancement in the form of recognizing new property values and qualifiers and integration of these into the main ontology structure to enhance it is a novel aspect of the present work.

Integration of a query-processing module capable of handling imprecise descriptions is also a new aspect of the system. The query-processing module performs imprecise query answering. Since exact matches for user-given concepts may not be found in documents, hence query-processing module employs fuzzy-reasoning principles to derive answers that may be imprecise matches for user-given concepts. The retrieved concepts are arranged according to their degrees of relevance to query concepts, where relevance is a function of values and qualifiers associated to query concepts and the concepts extracted from text documents. The efficacy of the system is established through experiments over several domains.

The rest of the paper is organized as follows. We review some related works on ontology-based text processing, fuzzy ontology structures and ontology enhancement in section 2. In section 3, an overview of the whole system along with its different modules is presented. Section 4 presents the document pre-processing process. Section 5 details the creation of fuzzy ontology structure. In Section 6 we have discussed the knowledge distillation process which extracts information components in an ontology-guided way. Section 7

presents an experimental evaluation of the knowledge distillation process. Section 8 explains the query processing mechanism and also presents several sample queries from different domains. Finally, we conclude and discuss future work in section 9.

2. Related work on text mining and Ontology-based text processing

Text mining refers to the process of extracting interesting and non-trivial patterns or knowledge from unstructured text documents. Text mining can provide an intelligent alternative to the current web querying process, which is mostly based on keyword searching. Recent research efforts in text retrieval encompass several areas including virtual database technology [1], web data modeling [12], wrapper generation [3,16,18], natural language processing based extraction [23], and ontology-based information extraction [6,9,26,27]. All these systems enable automatic extraction of relevant information from text documents. However, the first three techniques are heavily dependent on the structure of a web page. Natural Language Processing (NLP) based techniques provide more intelligent technology, but then understanding natural language texts in an unconstrained scenario is a complex problem by itself and is an active research area. Given the current state of research in NLP, combination of natural language processing principles within a structured framework promises to be more effective.

Tan [2] has proposed a general framework for text mining consisting of two components: *Text refining* that transforms free-form text documents into an *intermediate form*; and *knowledge distillation* that deduces patterns or knowledge from the intermediate form. Nahm and Mooney [31] have proposed the integration of text mining and information extraction approaches to deal with the information stored in unstructured text documents. Their system called Discovery from Text Extraction (DiscoTEX) is first trained to extract information and transform text into more structured data, which is then mined for interesting relationships. But the system does not use any domain knowledge due to which the proposed approach has problems when the same extracted entity or feature is represented by similar but not identical strings in different documents.

The use of ontology encourages enterprises to participate in knowledge-interchange by subscribing to a common, shared vision of entities and activities in a given domain [20,21]. Ontology based approaches are inherently resilient to the format of a web page and can be easily adapted to work for web pages from many distinct sources belonging to the same application domain. The use of ontological models to access and integrate large

knowledge repositories in a principled way has an enormous potential to enrich and make accessible unprecedented amounts of knowledge for reasoning [11].

In the next sub-section we will review some of the recent research efforts that have been directed towards the problems of ontology learning and design of ontology-based text processing systems.

2.1. *Ontology-Based text processing and Ontology learning*

Ontology is a conceptualization of a domain into a human understandable, but machine-readable format consisting of entities, attributes, relationships and axioms [17]. Ontology uses classes to represent concepts and supports taxonomy and non-taxonomy relations between classes. Ontology can be represented in one of the many languages like OIL [4], OWL⁵, XML⁶ etc. Since ontology describes a domain of interest in an unambiguous way, ontology-based text document processing schemes can help in alleviating a wide variety of natural language ambiguities present in a given domain.

Andreasen et al. [27] have proposed a system for content-based querying of texts based on the availability of an ontology that describes domain concepts. In their system the retrieval of text passages is based on matching descriptors from the text against descriptors from the noun phrases in the query using taxonomic reasoning with sub- and super-concepts. Snoussi et al. [9] have proposed an ontology-based approach that facilitates the formalization and the extraction of data from different sources. The extracted data is converted into a coherent structure so that users and agents can query them regardless of their origin.

Ontology learning helps in induction of new concepts and concept descriptors into the existing ontology as novel uses of the domain concepts are encountered while gathering information. Luke et al. [25] have proposed SHOE, a set of Simple HTML Ontology Extensions, which allow World Wide Web authors to annotate their pages with semantic knowledge. The annotations are used by a web crawler, which implements a graph traversal algorithm to discover related web pages. The performance depends heavily on the quality of annotation. Velardi, Fabriani, and Missikof [21] suggest a scheme for enhancing existing ontological structures with new information extracted from texts. Their work is based on identification of three primary kinds of concepts: *actor* which defines a relevant entity of the domain and is able to activate or perform process, *object* which is a passive entity on which a process operates, *process* which is an activity aimed at the satisfaction of an actor's goal.

Secondary concepts include *information components*, which are clusters of information pertaining to the information structure of an actor or an object, *information elements* which are atomic information elements that are parts of an information component and *elementary action* which denote activities that constitute process components and are not further decomposable. Based on the above definition of primary and secondary concepts, candidate terminological expressions are captured using a host of techniques ranging from stochastic methods to more sophisticated syntactic approaches. The concept forest generated is then manually integrated into a hand-crafted upper level ontology.

Hahn and Marko [30] introduce a dual-use methodology for learning both grammar and ontologies. This system automates the maintenance and growth of knowledge sources that are crucial for natural language text understanding—background knowledge of the underlying domain, linguistic knowledge about the lexicon, and the grammar of the underlying natural language. Learning occurs simultaneously with the ongoing text understanding process. The knowledge assimilation process identifies concepts based on linguistic analysis and assesses them for quality based on evidence underlying the generation. On the basis of the strength of evidence, hypotheses are ranked according to qualitative plausibility criteria, and the most reasonable ones are selected for assimilation.

Little, Hewett, and Embley [26] have proposed an ontology-based data extraction system, which uses an application ontology that describes a data-rich, ontologically narrow domain in a conceptual fashion. With inputs from a domain knowledge facilitator who can provide the knowledge for creating application ontology in an appropriate format, the system automatically generates a single wrapper that can be applied to any page relevant to the application domain.

Shamsfard and Barforoush [14] have suggested an ontology building approach in which the system starts from a small ontology kernel and constructs the ontology incrementally through text understanding. The kernel contains the primitive concepts, relations and operators to build an ontology. This model uses dynamic categories to handle changes and floating categories to handle multiple viewpoints and is implemented to extract information from natural language texts comprised of simple Persian (Farsi) sentences.

Li and Zhong [32] have described a methodology for ontology learning over an XML ontology scheme. The original ontology is in XML and is extended by using a list of facts and the frequency of their occurrences, which are provided by the users. Each fact supplies an individual opinion that specifies which class in the ontology the fact belongs to. A mass distribution of user profiles on the

⁵ <http://www.w3.org/TR/owl-ref/>

⁶ <http://www.w3.org/XML/>

ontology is used to incorporate an information object into the ontology.

Due to an increased interest in Bioinformatics, the sizes of Biomedical information repositories have gone up tremendously over the last decade. Ontologies play a very important role in providing efficient and focused search from these repositories. Several Biological ontologies like Gene Ontology, GENIA, TAMBIS ontology etc. have been built for this purpose. Consequently a lot of attention has been given towards ontology based processing of biomedical texts to enable researchers to search for relevant information components from a vast collection. Some of the successful ongoing projects in this area are the GENIA project, TEXTPRESSO [8], etc. Stevens et al. [22] have described how a Bioinformatics ontology can be built using OIL. They have also proposed Transparent Access to Multiple Bioinformatics Information Sources (TAMBIS), a mediation system that uses the TaO ontology to enable biologists to ask questions over multiple external databases using a common query interface.

2.2 Fuzzy Ontology structures

Since concept descriptions cannot be unambiguous, creation of Fuzzy ontology structures have also received a lot of attention in recent times. Widyantoro and Yen [5] have shown how fuzzy membership values associated to ontology concepts, along with a concept hierarchy, can be used for intelligent text retrieval. Starting with a set of manually tagged abstracts of papers from several IEEE Transactions, a fuzzy ontology is built on the collection of keywords. The abstracts are tagged based on their title, authors, publication date, abstract body, and author supplied keywords. The hierarchical arrangement of the terms in the newly generated ontology is dependent on their co-occurrence measures. The drawback of this system is its dependence on user judgment about the relevance of articles to user queries.

Wallace and Avrithis [15] have extended the idea of ontology-based knowledge representation to include fuzzy degrees of membership for a set of inter-concept relations defined in an ontology. The membership of these relations are used to judge the context of a set of entities, the context of a user and the context of the query for the purpose of intelligent information retrieval. A fixed set of commonly encountered semantic relations have been identified and their combinations are used to generate

fuzzy, quasi-taxonomic relations. This system lacks generality.

Quan, Hui, and Cao [29] have proposed an automatic fuzzy ontology generation framework – FOGA. They have incorporated fuzzy logic into formal concept analysis to handle uncertainty information for conceptual clustering and concept hierarchy generation. However, the quality of clustering is dependent on assignment of meaningful labels to initial class names, attributes and relations. This is done manually and requires domain expertise. This system is also not designed to extract fuzzy relational concepts from unstructured or semi-structured text documents.

2.3 Features of the proposed framework

The proposed system for information extraction and imprecise query answering is visualized as a complete ontology-aided knowledge management system, which has two main functions – *ontology based text pre-processing and knowledge distillation* and *imprecise query processing*. Text documents are pre-processed using shallow parsing techniques to generate semi-structured records. The knowledge distillation process performs ontology-based scanning of these records to extract information components. The information components are stored in a structured knowledge base for processing queries. Query processing is based on fuzzy reasoning. The structured knowledge base acts as a global repository for combining and collating information collected from multiple knowledge sources. Users need not be aware of the heterogeneity of the sources and query the collection using a common interface. This facility combines the ease of using a search engine with the capability of text mining for retrieving information embedded in texts.

The proposed system is capable of integrating new values and qualifiers extracted by the knowledge distillation process into the fuzzy ontology structure. None of the fuzzy ontology structures discussed earlier address the issue of incorporation of new values. Adding more values and qualifiers into an existing ontological structure helps in retaining the fixed structure of ontology but at the same time extends its capability to act as intelligent filters by progressively increasing its knowledge base. Thus the system has the facility to increase its lexicon or set of values and qualifiers for improving retrieval performance over a domain with time.

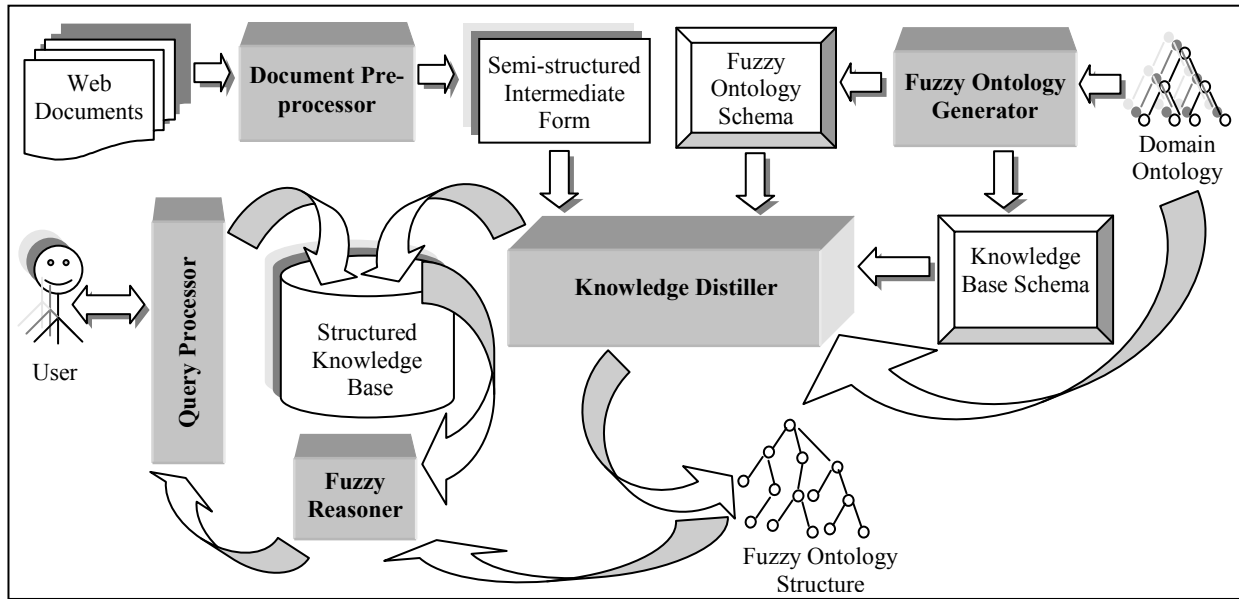


Fig. 1. System architecture

The fuzzy ontology structure is a novel structure that is created as an extension of traditional ontology structures. The novelty lies in describing concepts as a collection of $\langle \text{property_name}, \text{property_value}, \text{qualifier} \rangle$ triplets, where qualifiers can be linguistic variables. This allows defining the property-value of a concept with varying degrees of precision. Variable precision helps in imprecise query processing, where a concept can be retrieved from text even if it does not match a user given query exactly, and its relevance can be computed as a fuzzy similarity to original concepts using fuzzy reasoning methods. The structure can be easily adapted to reflect strength of association also, which may be a numeric value. Hence this structure is more general than the fuzzy ontology structures discussed earlier since this can accommodate both linguistic variables and numeric values.

3. System Architecture

This section outlines the complete architecture of the proposed system for ontology-based information extraction and imprecise query answering from text documents. The system, shown in figure 1, consists of five main modules – *Document Pre-processor*, *Fuzzy Ontology Generator*, *Knowledge Distiller*, *Query Processor*, and *Fuzzy Reasoner*. The functionalities of the modules are stated here briefly.

- The *Document Pre-processor* accepts free-form text documents and identifies information components by dividing them into individual record-size chunks after cleaning the Meta Language (ML) tags. This module

uses a Parts-Of-Speech (POS) tagger that assigns parts-of-speech to individual words. The identified information components along with their corresponding POS tags are stored in a semi-structured form for the use of the other modules.

- The *Fuzzy Ontology Generator* extends an existing ontology structure to a fuzzy ontology structure, by incorporating fuzzy classes into the existing ontology. Fuzzy classes are defined using multiple inheritances from values and qualifiers. The fuzzy ontology structure is then used to create the schema for a structured knowledge base that will store the information extracted from the texts.
- The *Knowledge Distiller* is responsible for extracting relevant concepts from the semi-structured intermediate records generated by the document processor, in an ontology guided way. The module uses a bi-directional inferencing mechanism for identifying relevant concepts. This module uses the domain concepts present in the ontology structure in a guided fashion to scan the record components and extract relevant information. The extracted concepts are used to populate the structured knowledge base and the fuzzy ontology structure appropriately.
- The *Query Processor* accepts user queries through an ontology-guided query interface and converts them into equivalent SQL queries. The SQL queries are passed on to the database engine, which extracts relevant instances from underlying structured knowledge base.

- The *Fuzzy Reasoner* implements fuzzy reasoning principles to reason with the extracted instances. Fuzzy reasoning is used to calculate the similarity between the concept descriptions present in user queries and those present in the extracted instances. A set of relevant instances, along with their degrees of relevance to the query concepts, is presented to the user.

The functional details of these modules are given in the following sections.

4. Document Pre-processor

The document pre-processor identifies and extracts segments from unstructured text documents. The processor consists of a Markup Language (ML) tags filter which removes the ML tags from a document before further processing. It then divides the document into individual record-size chunks, and stores them as unstructured records. Each record consists of a collection of sentences that are a part of the same paragraph, where a sentence termination is identified by the occurrence of a full stop.

Services Division (SIS) of the National Library of Medicine⁷ (NLM). Concept names are usually nouns, concept descriptors are adjectives, and description qualifiers mostly consist of adverbs. Thus our aim is to extract words with these POS tags from sentences since these can possibly contain imprecise concept descriptions.

The entire sentence is first divided into segments on the basis of stop words like commas, semicolons, conjunctions, etc. These segments are thereafter stored in a ternary tree. Other than the first segment in a sentence, wherein a noun is searched for, a segment is incorporated into the tree provided it has at least one adjective tag since this word is likely to contain a property value. If the first segment has at least one adjective tag it is added as a sub-tree in the document tree otherwise it is merged with the next segment having adjective tag(s). The ternary tree structure is defined as follows:

```
Structure Tree {
    String *Value;
    Struct Tree *Lchild;
    Struct Tree *Mchild;
    Struct Tree *Rchild;
}
```

Each sentence of a document and thereby the whole document is converted into an instance of the tree by

Dolcetto is a red table wine which is quite dry and has a slightly fruity flavor.
N X T J N N P X A J C X T A J N
Jordan: mostly arid desert; rainy season in west.
N A J N J N R N

Fig. 2. Texts with POS tags

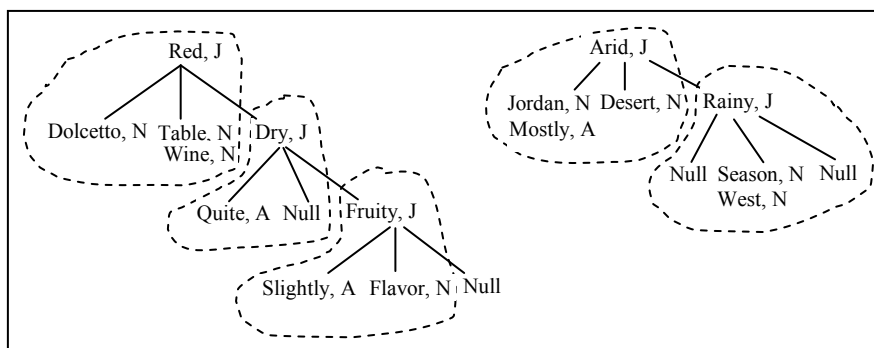


Fig. 3. Sample Ternary tree structures created from text documents

Parts-Of-Speech (POS) plays an important role in information extraction. The records are subjected to POS analysis using a POS tagger. The tagger assigns a POS tag to each word in a sentence. We have used a Tagger that has been developed by the Specialized Information

distributing the tags in the following way:

⁷ <http://tamas.nlm.nih.gov/taggercgi.html>.

Root (**R**): A node that contains the right most adjective word of a segment.

Lchild (**L**): A node that contains all the extracted words that are to the left of the word considered at R.

Mchild (**M**): A node that contains all words that are to the right of the word considered at R

Rchild: points to the root of the sub-tree constructed from the next segment.

The equivalent context-free grammar for this is given as follows:

Document (D) \rightarrow LRMD | ϵ

L \rightarrow (N+P+A+V+J)*

R \rightarrow J

M \rightarrow (N+P+V)*, where N, P, A, J, and V denote nouns, pronouns, adverbs, adjectives and verbs respectively.

Figure 2 shows two sample sentences, one describing a wine, and another describing the climate of a country, with POS tags assigned to the words. Figure 3 illustrates the ternary tree structures created from these tagged sentences. Multiple sentences of the same paragraph are linked through the lowest, rightmost child. This structure will be used by the *Knowledge-Distillation* process described in the next sub-subsection.

5. Fuzzy Ontology Generator

It has been established in the earlier sections that in some cases, concepts are best described through the use of imprecise property descriptors rather than through a <property_name, property_value> kind of structure. Incorporating imprecision into the ontology structure itself can help in resolving ambiguities arising due to differences in user requirement specification and concept descriptions embedded in text documents.

We propose a framework in which a property value can be specified to various degrees of precision using additional qualifiers. Thus in the proposed framework, an object description comprises of <property_name, property_value, qualifier>. Our aim is to extract elements from domain documents that fill up the slots in this triplet and provide valid descriptions of domain objects.

To accommodate imprecise concept descriptions, the fuzzy ontology structure uses modified concept descriptor classes. The fuzzy ontology structure contains two generic classes - a "Value" class and a "Qualifier" class. For each property descriptor class in the original ontology structure, two sub-classes are included in the fuzzy ontology structure - a "PropertyValue" class and a "PropertyQualifier" class, which are subclasses of the "value" and "qualifier" classes respectively. A qualifier class is constrained to have a collection of linguistic qualifiers. A set of linguistic qualifiers can be modeled as a graded set. The qualifier class along with its value is used to describe the property of a concept with varying degrees of precisions. A property value can also be

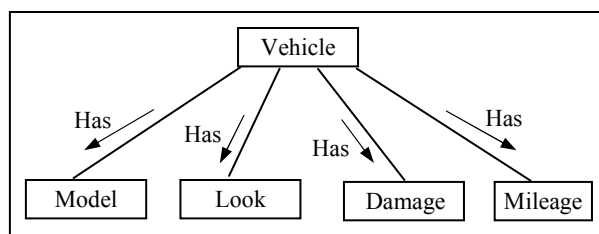


Fig. 4. A partial ontology for a Vehicle Description

associated with a NULL qualifier. In the fuzzy ontology structure a *FuzzyProperty* class is created through multiple inheritances from the value class and the qualifier class.

In order to illustrate the usefulness of the proposed ontology structure, consider the following descriptions picked up from a web document, which contains information about used vehicles put up for sale.

- *2000 Porsche Boxster: This car just in light damage on a 2000 model Porsche Boxster. Running and driving REPO clean title, very light damage (see photos).... parts needed are low cost and available, very easy repair.... this car packed w/ potential..the perfect repairable Porsche...call now.*
- *2000 Ducati 900 SS: This super clean 2000 Ducati 900 Super Sport looks like it's never even been dirty ... only 4100 miles, has front and light left side damage (minor), priced to sell at \$2775.*

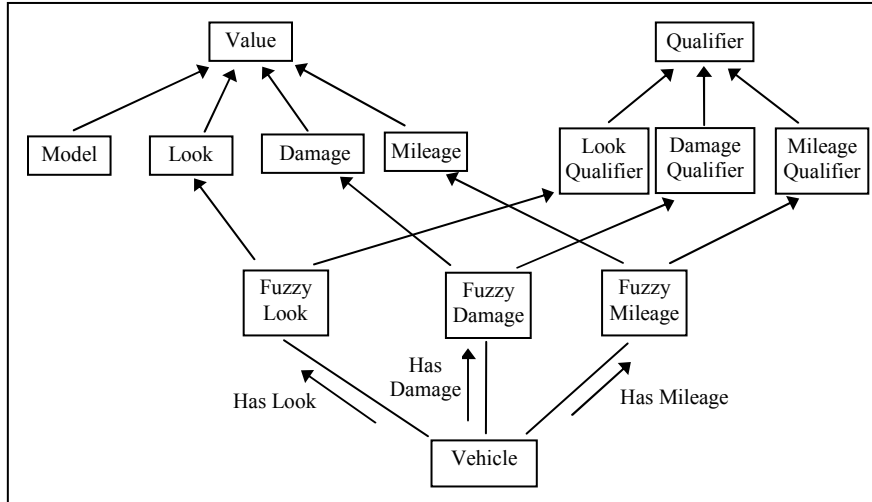


Fig. 5. Fuzzy property descriptors for the car properties shown in Fig. 4

The way these descriptions would be accommodated with usual property descriptors and the proposed fuzzy property descriptors, are illustrated in Figures 4 and 5 respectively. Figure 4 shows a partial view of property-descriptors to represent a “Vehicle”. Using this structure the *Damage* conditions of the “2000 Porsche Boxster” and the “2000 Ducati 900 SS” can be defined in one of two ways. In the first case both of them can be defined as “light”. In the second case, one can be defined as “very light” and the other as “light”. With the first representation, the difference in the degree of damage, if any, is missed. With the second representation, their similarity of being damaged to similar extents with very little difference is not captured. However, using the structure of Figure 5, the value of damage for both the vehicles will be stored as “light”, though the damage qualifier values will be different. For the first car it will be “very”, and for the second one it will be “Null”. This representation unambiguously captures the essence of both the vehicles being slightly damaged, with the second

one having slightly more damage than the first one.

Table 1 represents the redefined constraints, for the properties used to describe vehicles. For example the first row states that the attribute *HasLook* of a vehicle will be at most one instance of the *FuzzyLook* class. Similarly other rows define constraints for other slots.

Once the fuzzy ontology structure is created, the fuzzy ontology parser parses it to create an SQL schema for the structured knowledge base. This is accomplished through embedded SQL statements. Fuzzy property values and fuzzy qualifiers are attributes in this table. The ontology parser generates the list of objects, relationships, and constraints, which provide a basis for mapping the relationships in the ontology and the table declarations in the SQL schema. It also provides the cardinality constraints on the relationships like one-one, one-many, and many-many. The general layout of a knowledge base structure that is generated from the fuzzy ontology structure is shown in table 2.

Table 1. Redefined constraints for the Vehicle slots

Template Slots of Vehicle description classes			
Slot name	Type	Allowed Values/Classes	Cardinality
<i>HasLook</i>	Instance	FuzzyLook	0:1
<i>HasDamage</i>	Instance	FuzzyDamage	0:1
<i>HasMileage</i>	Instance	FuzzyMileage	0:1

Table 2. Structured knowledge base schema

Entity Name	Property-1 Qualifier	Property-1 Value	Property-2 Qualifier	Property-2 Value	...	Property-n Qualifier	Property-n Value
-------------	----------------------	------------------	----------------------	------------------	-----	----------------------	------------------

6. Knowledge Distiller

This module uses the ternary tree representation of the documents along with parent domain concepts present in the ontology structure, to populate the fuzzy ontology structure as well as the structured knowledge base with information extracted from web documents. Some of the key behavioral features of the instance generation mechanism are:

- (i) A particular object may have been described in a document by using some or all properties present in the ontology structure or by some other values, which are not present in the ontology structure.
- (ii) A document may or may not use the property name in conjunction with the property values for describing a concept. For example, in the document – *Roussanne is a light bodied, light red and very sweet wine from France's Loire Valley, often blended with Merlot*, though the property name *body* is mentioned explicitly; the property descriptors *taste* and *color* appear only implicitly through their values.

Guided by these observations, we have employed a two-pronged approach to populate the Fuzzy ontology structure. Given a property name, the instance generator looks for values to fill up the object description. This method allows accommodating object descriptions with property values that are not present in the underlying ontology. In the absence of a property name, a property value from the underlying ontology is used as a pointer to fill up the particular property slot.

Algorithm *Knowledge-Distillation* implements this approach to populate the knowledge base. This algorithm accepts as input the Fuzzy ontology structure schema, ontological entities, the ternary tree structure generated from the parsed documents, and the structured knowledge base schema as input. The output of this algorithm consists of information components, which are used to fill

up the slots of the Fuzzy ontology structure as well as the structured knowledge base. Detailed discussion on each step of the algorithm follows.

Algorithm: Knowledge-Distillation

Input: Root node **R** of the tree structure generated from document; Fuzzy ontology structure schema; List of ontological concepts and their relationships, Structured knowledge base schema generated by Ontology Parser

Output: Instances of concepts for populating fuzzy ontology structure and structured knowledge base.

Step1: Search the left child **L** of **R** for the *entity name* and put it in the column *entity_name* of the knowledge base.

Step2: Search the middle child **M** for a *property name*. When a property name explicitly appears in the document, it occurs either as a noun tag or a verb tag. Since a property name may or may not be explicitly present, the following two cases have to be considered.

Case 1: If Property name found – In this case, the value can be used directly.

Extract the value appearing in the value field of the root node and store it in the corresponding *property_value* column of the knowledge base as well as in the corresponding slot of the Fuzzy ontology structure, and go to step 4.

Case 2: If Property name not found – In this case, the *value* appearing in the value field of the root node can be used to identify the property.

Search the ontology structure to determine whether the value is present in the *property-value* set of a property. Since a value or a qualifier may be associated to more than one property, we use the property with which it has maximal co-occurrence.

Algorithm Knowledge-Distillation (ROOT)

Input: Ternary tree structure generated from Web documents (ROOT is the pointer to the tree structure), List of objects and relationships, fuzzy ontology structure schema, and structured knowledge base schema

Output: Instances of the Fuzzy Ontology structure and structured knowledge base.

Steps:

```
1. Ptr = ROOT // Start from root node
2. Entity_Name=NULL
3. If (Ptr ≠ Null) // If the tree is non-empty
   Property_Name=NULL; Property_Value=NULL; Qualifier_Value=NULL;
   a. Property_Name = SEARCH_PROPERTY_NAME (Property_name_list, Ptr -> MChild) // Search property name in the middle
   // child node.
   If Property_Name is not Null // Property name is explicitly mentioned in the document and found.
     Property_Value = Ptr -> Value // Value field of the root node is a property-value of the property stored in Property_Name
     Go to step 3 (b) // Proceed to search qualifier value
   Else // Property name is not explicitly mentioned in the document
     Property_Name = SEARCH_PROPERTY_VALUE (Property_Value_Lists, Ptr -> Value) // Search for a match of the Ptr
     // value field in the property value sets. In case of a valid match, return the corresponding property name.
     If Property_Name is not Null // The value at root node is a valid property value
       Property_Value = Ptr -> Value // Assign the content of Ptr value field to Property_Value.
       Go to step 3 (b) // Proceed to search qualifier value
     Else // The value at root node is not a valid property value
       Property_Value = SEARCH_PROPERTY_VALUE (Property_Value_Lists, Ptr -> LChild) // Search the Property value in
       // the left child node.
       If Property_Value is Null // The sub-tree does not have any property value.
         Go to step 3 (e) // Proceed to search the next sub-tree for property value and qualifiers (if any).
       End if
     End if
   End if
   b. Qualifier_Value = SEARCH_QUALIFIER_VALUE (Ptr -> Lchild) // Search qualifier value in the left child of the Ptr node.
   c. Instantiate the respective classes of the Fuzzy Ontology structure with Property_Value and Qualifier_Value.
   d. If Ptr=ROOT // The subtree rooted at ROOT node of the tree is under consideration.
     Entity_Name = SEARCH_ENTITY_NAME (ROOT -> Lchild) // Call this function exactly once
     Store Entity_Name, Property_Value, Qualifier_Value into Knowledge base // If entity name already exists in the knowledge
     // base, update only those fields that have Null values.
   Else // Remaining sub-trees are under consideration
     Store Property_Value, Qualifier_Value into Knowledge base // Update only those fields that have Null values.
   e. Ptr = Ptr -> Rchild // Proceed for the next sub-tree
   f. Go to step 3. // Repeat the above process for the next sub-tree
4. End if
```

Fig. 6. Knowledge-Distillation algorithm

Case 2.1: The value is present in the property-value set of some property - In this case extract the value appearing in the *value* field of the root node and store it in the corresponding *property_value* column of the knowledge base as well as in the corresponding slot of the Fuzzy ontology structure, and go to step 4.

Case 2.2: The value is not present - In this case go to step 3.

Step 3: Search the left child **L** for a property-value

Case 1: A property-value is found - In this case extract the found value and store it in the *property_value* column of the knowledge base as well as in the corresponding slot of the Fuzzy ontology structure and go to step 4.

Case 2: No property-value found - In this case the sub-tree under consideration is assumed not to contribute

any value for the Fuzzy Ontology structure. Go to step 6.

Step 4: Search the left child **L** of **R** for a qualifier. The qualifier for a value is likely to appear in this as an adverb or adjective tag. This node may have more than one qualifier so, the search proceeds from right-to-left. Hence the first qualifier to the left of a value found earlier is accepted as the qualifier for the value. If a valid qualifier is found, extract and store it in the corresponding *Qualifier_value* column of the knowledge base as well as in the corresponding slot of the Fuzzy ontology structure.

Step 5: If any of the above steps have yielded a match, block the matched *property-value* set from further search in the tree under consideration.

Step 6: Follow the right child pointer and replace **R** by it to consider the next sub-tree and repeat steps 2-6 until the value of the right child pointer is NULL.

In order to build a sound knowledge base, a document collection should be used, where all documents belong to the same domain. The above steps are applied to all documents in the collection. In order to decide the correct class for new qualifiers and values extracted, we have applied statistical analysis on the learned value and qualifier sets independently. For all unique values in the set, frequencies of their occurrences with different properties are computed and a value is assigned to the property with which it has maximum number of occurrences. The same is done for the assignment of qualifiers to different properties. The above steps are applied iteratively till the state of the knowledgebase stabilizes. Thus the Knowledge Distiller module can start with a seed ontology, which contains a small set of property values and qualifiers, and then iteratively accumulate new values and qualifiers from text documents.

Figure 6 presents the algorithm formally.

7. Experimental Results for text processing and Ontology enhancement

The overall system performance is dependent on the performance of two units - the Knowledge distiller and

the Query Processor separately. We have evaluated the two components separately and provide here the experimental details of evaluation. In this section we will present a performance analysis of the knowledge distillation process. To evaluate this module, we have collected documents from the Web and then evaluated the effectiveness of the distillation process in identifying concept descriptors from these. The evaluation process judges the effectiveness of the method in extracting both old and new values and qualifiers from domain documents. We have provided performance analysis over four domains. For analysis of wine documents, we have used the wine ontology developed by W3C and extracted information from a large collection of web documents to enrich it further. For the other three domains, we started with seed ontologies that contain property names and some values. We created initial ontology structures for these domains using Protégé⁸, and then enhanced these using the values and qualifiers extracted through text mining. Protégé is an integrated software tool used by system developers and domain experts to develop knowledge-based-systems. It may be noted that all qualifier sets associated with the fuzzy ontology structure are initially empty and gets populated with qualifiers extracted from the text sources.

Section 7.1 presents the nature of the fuzzy ontology

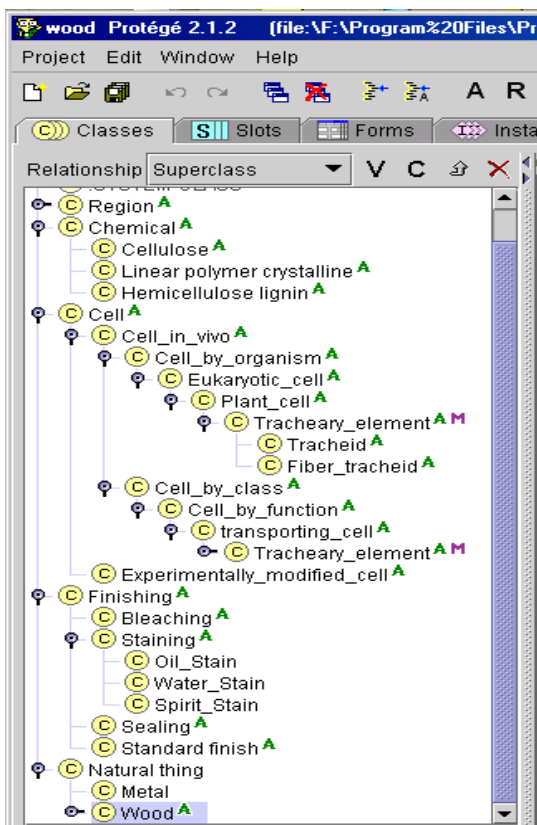


Fig. 7. Wood Ontology structure

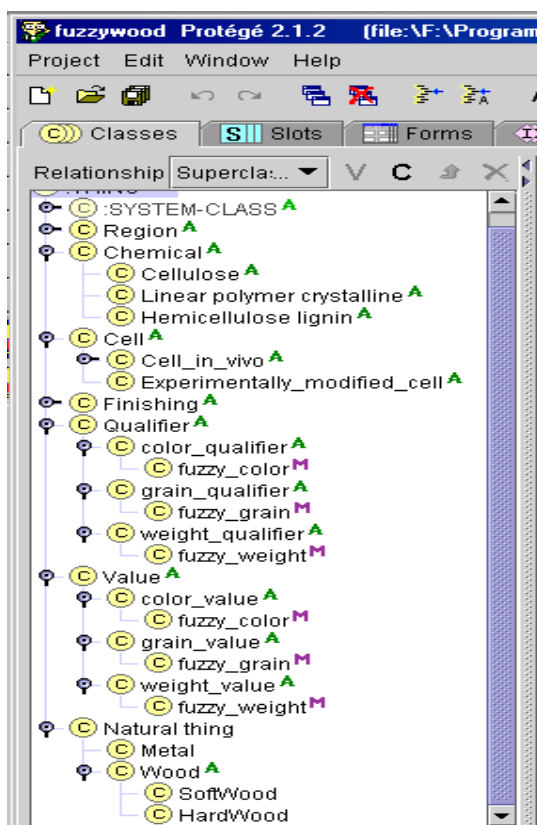


Fig. 8. Fuzzy Wood Ontology structure

Instances of <i>fuzzy_color</i> class		Instances of <i>fuzzy_weight</i> class	
Color Qualifier	Color Value	Weight Qualifier	Weight Value
Heavy	Black	Moderately	Heavy
Dark	Brown	Very	Heavy
Golden	Brown	Moderately	Light
Light	Brown	Very	Light
Rich	Brown	Null	Medium
Null	Creamy		
Dark	Gray		
Deep	Orange		
Rich	Orange-red		
Light	Pinkish-brown		
Dark	Purple-brown		
Bright	Red		
Deep	Red		
Medium	Red-brown		
Rich	Red-brown		
Dark	Reddish		
Light	Reddish-brown		
Pale	Yellowish		

Instances of <i>fuzzy_grain</i> class	
Grain Qualifier	Grain Value
Null	Curly
Null	Distinctive
Fairly	Even
Fine	Even
Very	Even
Null	Medium
Very	Pronounced
Generally	Straight
Usually	Straight

Fig. 9. A partial list of values and qualifiers extracted from Wood documents

structures for the various domains and illustrates the instantiation of these with information extracted from documents. In section 7.2 we provide an evaluation of the distillation process in terms of precision and recall. Later, we will illustrate how concept descriptions including the imprecise descriptors extracted from web documents are used for answering user queries. This is presented in the next section in which we have shown some query processing examples over these domains to illustrate how the user gets information about wines, wood, vehicles or weather even with imprecise descriptions about the concepts or qualities of the desired entity.

7.1 Fuzzy Ontology creation and instantiation

To start with, an ontology is described as a collection of a root concept, property descriptor concepts and relations describing associations among these concepts. Since the wine⁹ ontology structure is already available, we will describe the process of creating a fuzzy ontology using another domain. Information on various types of wood are stored in this new ontology, which we call the wood ontology. The wood ontology uses three properties for describing wood, with the value sets initialized as follows:

- {Cream, Red, Brown, White, Yellow, Orange, Gray, Black, Pink} for *color* property
- {Straight, Open, Close, Even, Curly, Wavy} for *grain* property, and
- {Light, Heavy, Medium} for *weight* property, and

Using these property descriptors, the fuzzy properties for this domain, created through multiple-inheritance are:

- *fuzzy_color*, which is a subclass of *color_value* class and *color_qualifier* class.
- *fuzzy_grain*, which is a subclass of *grain_value* class and *grain_qualifier* class.
- *fuzzy_weight*, which is a subclass of *weight_value* class and *weight_qualifier* class.

Now, any instance of wood would be described by these three properties using varied degrees of precision. Hence *color* slot of class wood is constrained to take its values as an instance of *fuzzy_color* class. Similarly other slots are constrained to accept their values as an instance from their respective fuzzy classes. The original and the redefined taxonomic structure of the wood ontology, its classes and constraints are shown in Fig. 7 and 8 respectively.

Fig. 9 shows the instances of wood descriptions extracted through knowledge distillation from a collection

⁹ <http://protege.stanford.edu/plugins/owl/owl-library/>

Instances of <i>fuzzy_look</i> class		Instances of <i>fuzzy_condition</i> class		Instances of <i>fuzzy_damage</i> class	
Look Qualifier	Look Value	Condition Qualifier	Condition Value	Damage Qualifier	Damage Value
Null	Beautiful	Null	Super	Very	Light
Very	Beautiful	Null	Fantastic	Null	Light
Very	Clean	Very	Fantastic	Null	Minor
Super	Clean	Very	Nice	Very	Extensive
Null	Great	Null	Nice	Very	Minor

Fig. 10. Partial list of values and qualifiers extracted from Vehicle documents

Instances of <i>fuzzy_climate</i> class		Instances of <i>fuzzy_temperature variation</i> class		Instances of <i>fuzzy_summer</i> class	
Climate Qualifier	Climate Value	Temp. Variation Qualifier	Temp. Var. Value	Summer Qualifier	Summer Value
Most	Moderate	Null	High	Null	Hot
Null	Tropical	Little	Seasonal	Very	Hot
Mostly	Temperate	Average	High	Null	Moist
Mostly	Tropical	Slight	Seasonal	Null	Cool
Mostly	Arid	Moderately	High	Always	Hot
Mostly	Semiarid	Null	Moderate	Generally	Hot
Null	Semiarid	Null	Seasonal	Null	Dry
Cold	Temperate	Relatively	Low	Intensely	Hot
Cool	Temperate	Constantly	High	Constantly	Hot
Mild	Temperate	Very	High	Mostly	Hot
Null	Subtropical	Severe	Low	Extraordinarily	Hot

Fig. 11. Partial list of values and qualifiers extracted from Weather documents

Instances of <i>fuzzy_flavor</i> class		Instances of <i>fuzzy_color</i> class		Instances of <i>fuzzy_taste</i> class	
FlavorQualifier	FlavorValue	ColorQualifier	ColorValue	TasteQualifier	TasteValue
Slightly	Bitter	Pale	Cherry-red	Slightly	Bitter
Zesty	Delicate	Ruby	Red	Fine	Dry
Null	Dry	Orange	Red	Light	Dry
Null	Exquisite	Indigenous	Red	Medium	Dry
Null	Favorite	Light	Red	Null	Dry
Fresh	Fruity	Robust	Red	Quite	Dry
Usually	Fruity	Premium	Red	Slightly	Flinty
Mellow	Nutty	Bright	Ruby-red	Null	Fresh
Smooth	Nutty	Tenuous	Straw	Full	Mellow
Null	Robust	Null	White	Medium	Sweet
Null	Spicy	Pale	White	Slightly	Sweet
Harmonious	Velvety	Bright	Yellow	Very	Sweet

Fig. 12. Partial list of values and qualifiers extracted from Wine documents

of wood documents collected from the Web. This figure shows some property-values that were not there in the ontology originally but extracted by the knowledge-distillation process. The set of qualifiers, which is entirely

new, are later on arranged into an ordered set through human intervention. Fig. 10 shows the set of values and qualifiers extracted by the knowledge distiller for the vehicle domain. Fig. 11 shows the values and qualifiers

extracted from documents on weather. Fig. 12 shows a partial list of values and qualifiers extracted from documents on wine. The point to note here is the existence of a number of new values that were distilled from the documents.

7.2 Performance analysis of the Knowledge Distiller

The quality of knowledge distillation can be measured by comparing its information extraction accuracy against human curation. Our intention was to study the applicability of ontology-guided precise and imprecise concept description extraction from general-purpose unstructured texts. We considered documents on four widely different domains - wine, vehicles, wood and weather as our test-bed. To build our corpora we have downloaded documents from the World Wide Web. Each of these documents describes one type of entity and contains around 10 sentences. Presently, the wine corpus has 500 documents consisting of 52,320 words. The wood corpus has 475 documents consisting of 55,590 words. The weather corpus contains 264 documents. The vehicle corpus is comparatively very small, and there are only 23 documents. We manually inspected these documents to build a complete compilation of all possible elements to be extracted and curated them under different categories (shown in column 1 of Table 3): entity names, property names, property values and property value qualifiers. The elements extracted by the Knowledge Distiller are automatically stored in different columns of the database depending on their types. It may be noted that property names do not appear explicitly in the database since the database attributes are created from the ontology structure itself. However, the recognition of property names is important since it affects the property-value recognition process and thereby the overall system performance. The performance of this module is computed using standard measures of *precision* and *recall*, which are defined as follows:

$$Precision = \frac{\text{Number of relevant elements extracted}}{\text{Total number of elements extracted}}$$

$$Recall = \frac{\text{Number of relevant elements extracted}}{\text{Total number of relevant elements actually present in the corpus}}$$

Table 3 summarizes these values for the different domains. As is observed, the precision of the system is quite high. This indicates that most of the extracted instances are correctly identified. However, the recall value of the system is somewhat low. This indicates that several relevant elements are not extracted from the text.

To analyze the performance further, we identify the causes of this problem as arising from three different, though not mutually exclusive, sub-tasks performed by this module, namely entity name extraction, property value extraction and property qualifier extraction. The problems associated to each of these recognition tasks are further analyzed as follows:

- *Named-entity recognition* - It may be observed from Table 3 that the recall and precision of recognizing names is very high in the case of weather and vehicle documents, though recall was comparatively lower for wine and wood documents. Precision is high for all the domains. This indicates that most of the names were correctly recognized for all domains. For wine and wood documents a few other names were also identified as entities, which lowered the precision slightly. These were mostly names of places where a specific wine is found or a wood is grown.

The reason for low recall values in certain domains was identified as follows. We observed that most errors occur when the name is composite in nature. For example, while the name *Verdlet* is easy to identify as a noun, it is not so for the name *Blush Niagara*. The Parts-Of-Speech tagger sometimes identifies *Blush* as an adjective. This problem is particularly severe for domains where standard naming conventions do not exist. Though vehicles also had composite names, due to the standard pattern followed for all vehicle documents, the recall for this domain was high. This problem can be overcome by employing a set of additional Entity-Recognition rules, which can take care of the peculiar naming conventions of the domain. This is the usual approach adopted for Biological document processors to deal with a large entity collection with rather inconsistent naming standards [7], [20].

Table 3. Performance Metrics for the Knowledge Distiller in terms of Precision and Recall

Domain	Type of Extracted Information	# Elements in Source	# Extracted Elements recognized correctly	#Incorrect Elements extracted	Precision	Recall
Wine	Named Entities (Wine names)	1092	971	42	95.85	88.92
	Property values	2172	1591	31	98.09	73.25
	Qualifier Values	1492	1126	185	85.89	75.47
Wood	Named Entities (Wood names)	965	856	49	94.59	88.70
	Property values	1962	1177	109	91.52	59.99
	Qualifier Values	650	486	82	85.56	74.77
Weather	Named Entities (Country names)	264	264	0	100.00	100.00
	Property values	617	440	13	97.13	71.31
	Qualifier Values	126	95	9	91.35	75.40
Vehicle	Named Entities (Vehicle names)	23	23	0	100.00	100.00
	Property values	101	92	1	98.92	91.09
	Qualifier Values	12	10	1	90.91	83.33

- Property value recognition* - A word is recognized as a property value if either it is a member of a property-value set in the ontology structure, or it is associated with a property name in the web document. For example, let us consider the following document along with the POS tags associated to the words:

```

Verdlet is a delicate, fruity, white table
N      X T  N      J      J      N
wine, with the pleasingly crisp, slightly
N      R  T      A      V      A
flinty taste.
J      N

```

In the above example, though the property name *color* does not explicitly appear in the description, the knowledge base is filled up correctly, since *white* is recognized as an adjective by the tagger, and it is also present in the property value set of *color* in the domain ontology. Though a new word, the description “flinty” is also correctly recognized as a *taste value* since it occurs explicitly in association to property name— *taste*. However, the word *fruity*, though assigned the correct parts-of-speech by the tagger, is not recognized as a *property value* for flavor by our system, since it is neither present in any *property value set* in the underlying ontology structure nor does it occur in association to any specific *property name* in the document.

- Property value qualifier recognition* – This task has relatively less precision and recall values than other tasks. There are two reasons that we have identified for this. Firstly, in many cases a property value may be associated with more than one qualifier. Since our system extracts only the one that is closest to the property value, a lot of qualifiers are missed. For example, consider the following sentence picked-up from the wine domain:

Chianti: a very bright ruby red color; a dry flavor, which becomes delicate...

Here we found that the color value of “Chianti” is associated with two different qualifiers *bright* and *very*. Our system extracts only *bright* as a qualifier value and associates it with the *color value* of wine. In order to reflect the true performance of the module, during performance evaluation we have counted *very* as a missed but relevant element. This problem can be easily tackled by considering adverb chains rather than single adverbs as qualifiers [24].

The second problem creeps in due to the fact that many adverbs which occur in the document are not really qualifiers, though they occur sufficiently close to some property value. An example of such a sentence is given below in which the word *occasionally* is wrongly judged as a qualifier for *taste* whose value is *slight*.

Valpolicella, an everyday red wine, . . .and
 N T J J N C
 has plenty of body, and occasionally a
 x N R N C A T
 slight taste of bitter almonds.
 J N R J N

8. Query Processor and Fuzzy Reasoner

In this section we will present the design of the *Query Processor* and *Fuzzy Reasoner* modules. The Query Processor accepts user queries through an ontology-guided query interface and converts them into equivalent SQL statements. The SQL queries are passed on to the database engine, which extracts relevant instances from the underlying structured knowledge base. The extracted instances are passed on to the Fuzzy reasoner module to calculate the similarity value between the concept descriptions posed by the user, and those present in the extracted instances. The final set of instances judged as relevant, along with their degrees of relevance, is presented to the user.

The overall task of query processing is a two-step process – acceptance and conversion of the user query into a corresponding SQL query and then extracting and finding the relevant answer from the structured knowledge base. The processing steps are explained in the succeeding subsections.

8.1 Accepting user query

The *Query Processor* uses the underlying Fuzzy ontology model and the structured knowledge base, which defines and stores the domain concepts using a set of property values and qualifiers. The query processor interface, shown in Fig. 13, guides the user to enter his/her queries in a given format, which is then converted into an SQL query. Queries are formulated through this

Fig. 13. I/O Interface

interface as follows:

- A user can frame queries by selecting the property and qualifier values from corresponding list boxes.
- A user can specify constraints on a property by selecting only a qualifier, or a value, or both. Pull-down menus containing lists of existing property names, property values, and qualifiers are provided.
- Complex queries can be formulated by selecting one or more of the logical operators AND/ OR/ NOT

Once a user query is accepted, a corresponding SQL query is built out of it using the property names, property values and logical operators (if any). The qualifiers are not considered for building the SQL query. They are considered later for judging the relevance of an answer.

8.2 Instance extraction and relevance computation

Since user given descriptions may not exactly match any known description in the knowledge base, the answer generation mechanism applies fuzzy reasoning to find good matches. The following steps explain the answer generation process:

Step 1: Formulation of SQL queries using user-given parameters – During this phase, only the given property names, property values and logical operators (if any) are used to formulate the SQL query. For example if the user enters the query – *List the wines with light yellow color or medium bitter taste*, the corresponding SQL query is generated as:

```
SELECT WineName, ColorQualifier, ColorValue,
TasteQualifier, TasteValue
FROM WineKnowledgebase
WHERE ColorValue = "yellow" or TasteValue =
"Bitter";
```

The SQL query is processed using the knowledge base and fuzzy ontology model.

Step 2: Precise query processing for finding exactly matching answers – In this step the SQL statements generated in step 1 are executed and the instances of knowledge base with exact value matches are extracted as intermediate results. The extracted instances for the earlier query are as follows:

WineName	Color Qualifier	Color Value	Taste Qualifier	Taste Value
Grignolino	Null	Null	Slightly	Bitter
Racioto di soave	Bright	Yellow	Null	Null

Step 3: Fuzzy similarity computation for all answers extracted – During this stage, the user given qualifiers are used to compute the similarity of each extracted instance

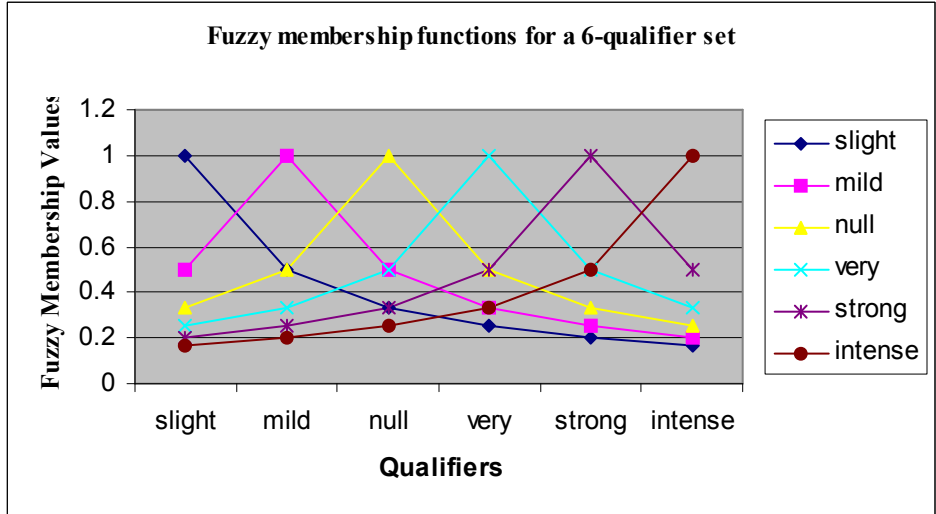


Fig. 14. Fuzzy membership functions for a 6-qualifier set

to the actual user query in order to compute the final set of answers.

To calculate the fuzzy similarity value between the concept descriptions given in the user query and those extracted from the knowledge base, we have redefined the concept of similarity measures discussed in [9]. If u and v are two objects in a given universe of discourse U , a similarity measure is a function: $SIM(u, v) \rightarrow [0, 1]$. In our framework, since the qualifier set for different properties is modeled as a graded set, the distance between two qualifiers in the collection reflects their degree of dissimilarity. The distance between the value v_i at position i and the value v_j at position j within a set is defined as:

$$d(v_i, v_j) = |i - j| \dots\dots\dots(i)$$

The similarity between two concepts u and v is computed as:

$$SIM(u, v) = 1 - \frac{d(u, v)}{MAX + 1} \dots\dots\dots(ii)$$

where MAX is the maximum distance between the concepts in the graded set.

This function satisfies the condition $0 < SIM(u, v) \leq 1$. Using this function each qualifier can be assigned a fuzzy membership value to another qualifier class. The membership values determined by this fuzzy membership function for a sample 6-qualifier set are shown in figure 14. However, other fuzzy membership functions can be also employed to assign the membership values to the qualifiers.

The similarity of an extracted instance to the user given description is computed in terms of the degree of

match between user-specified property values and qualifiers to those appearing in the knowledge base.

Let the user given description for each property P be represented by $C = \langle P, V, Q_1 \rangle$, where V denotes the property value and Q_1 denotes the qualifier. Let x be a retrieved instance which has value V for property P and qualifier Q_2 . Let $\mu_C(x)$ denote the similarity between the instance x and the user given description C for property P . $\mu_C(x)$ is computed as follows:

Case 1: Both Q_1 and Q_2 are present in the qualifier set of P in the fuzzy ontology – In this case, $\mu_C(x) = SIM(Q_1, Q_2)$.

Case 2: $Q_1 = NULL$ or $Q_1 = Q_2$ – The first situation arises when user does not select/enter a qualifier with a property value. In this case, $\mu_C(x) = 1$. That is, all instances with matching values are accepted as exact matches. Similarly, if there is an exact match between the user given concept descriptions and those present in the retrieved instances, the similarity value $\mu_C(x) = 1$.

Case 3: Either Q_1 or Q_2 or both are not present in the qualifier set and they are not $NULL$ – In this case, it is judged that the query processor is not able to resolve the differences in the descriptions, and so $\mu_C(x) = 0$.

For complex queries i.e. queries in which multiple property descriptions C_1, C_2, \dots, C_n have been specified by the user and are combined with logical AND, OR, and NOT operators - the final similarity value is computed by using fuzzy intersection, union and negation operations respectively. The membership computation functions for intersection, union, and negation of fuzzy concepts C_1, C_2, \dots, C_n are defined as follows:

$$\mu_{C_1 \cup C_2 \cup \dots \cup C_n}(x) = MAX[\mu_{C_1}(x), \mu_{C_2}(x), \dots, \mu_{C_n}(x)]$$

$$\mu_{C_1 \cap C_2 \cap \dots \cap C_n}(x) = \text{MIN} [\mu_{C_1}(x), \mu_{C_2}(x), \dots, \mu_{C_n}(x)]$$

$$\mu_{\neg C_1}(x) = 1 - \mu_{C_1}(x)$$

Continuing with the earlier example, the similarities of the extracted instances with the user given description are given in the following table. It may be noted that the similarity values are based on either a color match or the taste match, through the qualifiers. The second instance has a lower degree of similarity than the first, since the pair of qualifiers *bright* and *light* for color differs more than the pair *slightly* and *medium* for taste.

WineName	Color Qualifier	Color Value	Taste Qualifier	Taste Value	Similarity Value
Grignolino	Null	Null	Slightly	Bitter	0.67
Racioto di soave	Bright	Yellow	Null	Null	0.31

Step 4: Answer generation – only those instances, which have similarities greater than or equal to a given threshold with the user-description, are presented to the user.

8.3 Illustration of imprecise query processing

We now present some sample queries and answers generated for them, using the knowledge bases that were generated during the earlier phase. These knowledge bases contain entity descriptions that are stored in a database, whose schema is decided by the underlying fuzzy ontology structure of the corresponding domain. Each retrieved entity is accompanied by a similarity value to indicate its relevance to the original query. For presenting a qualitative performance analysis of our system, in each of the following tables, we have also shown those instances which have property value matches with the query, but are ultimately judged to have similarity value zero. Results of ten queries are presented here from among which, first two are from the wine domain, next three each from the domains of wood and weather, and then two from the vehicle domain.

Query 1: List the wines with medium sweet taste.

The corresponding SQL query is generated for retrieving relevant entities from the knowledge base, which are then subjected to relevance computation:

```
Select WineName, TasteQualifier, TasteValue
From WineKnowledgebase
Where TasteValue = "Sweet";
```

Result after fuzzy relevance computation:

Wine Name	Taste Qualifier	Taste Value	Similarity Value
Niagara	Medium	Sweet	1.00
Blush Niagara	Medium	Sweet	1.00
Asti Spumante	Slightly	Sweet	0.67
Red Sparkler	Slightly	Sweet	0.67
White Sparkler	Slightly	Sweet	0.67

Wine Name	Taste Qualifier	Taste Value	Similarity Value
Gewurztraminer	Slightly	Sweet	0.67
Brachetto	Null	Sweet	0.56
Muscat Di Tanta Maria	Null	Sweet	0.56
White Muscadine	Null	Sweet	0.56
Blush Muscadine	Null	Sweet	0.56
Alpine Burgundy	Null	Sweet	0.56
Red Muscadine	Null	Sweet	0.56

Query 2 exemplifies the use of the logical operator AND. In the answer set for this query, though the first instance has more matches with the user given description for one property, due to the use of MIN function for fuzzy intersection operation, its ultimate relevance goes down. It may also be noted that presently the system is incapable of judging the similarity between “slightly” and “fresh”, since they cannot be judged on the same scale. Hence the last two instances are inferred as irrelevant, which should not be ideally so. This aspect needs further analysis.

Query 2: List the wines with slightly sweet taste and slightly fruity flavor.

```
Initial SQL> Select WineName, TasteQualifier,
TasteValue, FlavorQualifier, FlavorValue
From WineKnowledgebase
Where TasteValue = 'sweet' AND FlavorValue =
'Fruity';
```

Results after fuzzy relevance computation:

WineName	Taste Qualifier	Taste Value	Flavor Qualifier	Flavor Value	Similarity Value
Gewurztraminer	Slightly	Sweet	Null	Fruity	0.50
White Muscadine	Null	Sweet	Null	Fruity	0.50
Blush Muscadine	Null	Sweet	Null	Fruity	0.50
Red Muscadine	Null	Sweet	Null	Fruity	0.50
Niagara	Medium	Sweet	Fresh	Fruity	0.00
Blush Niagara	Medium	Sweet	Fresh	Fruity	0.00

Following are some more queries, their SQL versions and the retrieved results from the other domains.

Query 3: List very heavy weighted woods.

```
Initial SQL> Select WoodName, WeightQualifier,
WeightValue
From WoodKnowledgebase
Where WeightValue = "Heavy";
```

Results after fuzzy relevance computation:

Wood Name	Weight Qualifier	Weight Value	Similarity Value
Oak	Very	Heavy	1.00
Hickory	Moderately	Heavy	0.67
Eucalyptus	Null	Heavy	0.33
Mahogany	Null	Heavy	0.33
Maple	Null	Heavy	0.33
Mesquite	Null	Heavy	0.33
Benge	Null	Heavy	0.33

It may be noted that in this example, an entity with more similarity to the user query is judged as more relevant. Hence Oak is definitely a better choice as *heavy* wood, than Eucalyptus or Maple etc.

Query 4 illustrates the use of the logical OR operator. Results show that though Birch is initially retrieved since it has curly grains, however it was later discarded because the qualifier “often” could not be compared with the query qualifier “fine”. It may be noted that Oak is once again a match since it is very heavy and though it does not have curly grains, user has specified OR of two properties. Hence any one exact match yields a value of 1.0. All other instances are imprecise matches for the query.

Query 4: List the woods that have very heavy weight or fine curly grains.

```
Initial SQL> Select WoodName, WeightQualifier,
WeightValue, GrainQualifier, Grain Value
From WoodKnowledgebase
Where WeightValue = "Heavy" OR GrainValue =
"Curly";
```

Results after fuzzy relevance computation:

Wood Name	Weight Qualifier	Weight Value	Grain Qualifier	Grain Value	Simil. Value
Oak	Very	Heavy	Null	Null	1.00
Hickory	Moderately	Heavy	Null	Straight	0.67
Mahogany	Null	Heavy	Null	Curly	0.5
Eucalyptus	Null	Heavy	Null	Null	0.33
Maple	Null	Heavy	Usually	Straight-grained	0.33
Mesquite	Null	Heavy	Null	Null	0.33
Benge	Null	Heavy	Null	Wavy	0.33
Birch	Null	Medium	Often	Curly	0.00

Query 5 also illustrates the use of logical OR operators with more properties in user query.

Query 5: List the woods that have either pale yellowish color or very heavy weight or wavy grain.

```
Initial SQL> Select WoodName, ColorQualifier,
ColorValue, WeightQualifier, WeightValue,
GrainQualifier, GrainValue
From WoodKnowledgebase
Where ColorValue= "yellowish" OR WeightValue=
"Heavy" OR GrainValue= "Wavy";
```

Results after fuzzy relevance computation:

Wood Name	Color Qual.	Color Value	Wt. Qual.	Wt. Value	Grain Qual.	Grain Value	Simil. Value
Oak	Gray	Brown	Very	Heavy	Null	Null	1.00
Honduras Mahogany	Null	Reddish	Null	Medium	Null	Wavy	1.00
Benge	Pale	Yellowish	Null	Heavy	Null	Wavy	1.00

Wood Name	Color Qual.	Color Value	Wt. Qual.	Wt. Value	Grain Qual.	Grain Value	Simil. Value
Cedar-White	Null	Yellowish	Null	Null	Null	Null	0.88
Hickory	Light	Reddish-brown	Moderately	Heavy	Null	Straight	0.67
Eucalyptus	Null	Pinkish-brown	Null	Heavy	Null	Null	0.33
Mahogany	Null	Null	Null	Heavy	Null	Curly	0.33
Maple	Null	Reddish-brown	Null	Heavy	Usually	Straight-grained	0.33
Mesquite	Null	Null	Null	Heavy	Null	Null	0.33

Queries 6, 7 and 8 are from the weather domain. The information is extracted from a document which contains description of climatic conditions for 264 countries.

Query 6: List the countries that have very hot summer and cool winter.

```
Initial SQL> Select CountryName,
SummerQualifier, SummerValue, WinterQualifier,
WinterValue
From WeatherKnowledgebase
Where SummerValue= "hot" AND WinterValue=
"cool";
```

Results after fuzzy relevance computation:

Country Name	Summer Qualifier	Summer Value	Winter Qualifier	Winter Value	Simil. Value
Kuwait	intensely	Hot	Null	cool	0.83
Bhutan	Null	Hot	Null	cool	0.67

Query 7: List the countries that are arid and very hot.

```
Initial SQL> Select CountryName,
ClimateQualifier, ClimateValue, SummerQualifier,
SummerValue
From WeatherKnowledgebase
Where ClimateValue= "arid" AND SummerValue=
"hot";
```

Results after fuzzy relevance computation:

Country Name	Climate Qualifier	Climate Value	Summer Qualifier	Summer Value	Simil. Value
Bahrain	Null	Arid	very	hot	1.00
Qatar	Null	Arid	very	hot	1.0
Kazakhstan	Null	Arid	Null	hot	0.25
Mali	Null	Arid	Null	hot	0.25

Query 8: List the tropical countries that have little seasonal variation.

```
Initial SQL> Select CountryName,
ClimateQualifier, ClimateValue,
Temp_variationQualifier, Temp_variationValue
From WeatherKnowledgebase
Where ClimateValue= "tropical" AND
Temp_variationValue= "seasonal";
```

Results after fuzzy relevance computation:

Country Name	Climate Qual.	Climate Value	Temp_variation Qualifier	Temp_variation Value	Simil. Value
American Samoa	Null	tropical	Little	seasonal	1.00
Antigua	Null	tropical	Little	seasonal	1.00
Aruba	Null	tropical	Little	seasonal	1.00
Cambodia	Null	tropical	Little	seasonal	1.00
Dominican Republic	Null	tropical	Little	seasonal	1.00
French Guiana	Null	tropical	Little	seasonal	1.00
Guam	Null	tropical	Little	seasonal	1.00
Johnston Atoll	Null	tropical	Little	seasonal	1.00
Northern Mariana Islands	Null	tropical	Little	seasonal	1.00
Puerto Rico	Null	tropical	Little	seasonal	1.00
Saint Kitts	Null	tropical	Little	seasonal	1.00
Saint Vincent	Null	tropical	Little	seasonal	1.00
Fiji	Null	tropical	Slight	seasonal	0.75
Papua New Guinea	Null	tropical	Slight	seasonal	0.75
Montserrat	Null	tropical	Null	seasonal	0.50

The documents that we collected on vehicles described old vehicles that were on sale. Owners described cars in terms of mileage, color, condition, damage etc. Some sample queries and answers generated from them are shown below. This domain is interesting from a different viewpoint as illustrated with query 9. Though in general a buyer will be happier with a “very clean” car when (s)he is looking for a “clean” one, the earlier relevance computation function did not take care of this. However, this is an exception and we have modeled this as such. In case the query was for a “very clean” car, then a “clean” car would be judged less relevant both by our system and the buyer! Query 10 shows some other properties with which the usual reasoning procedure works well.

Query 9: List vehicles that have clean look and no damage.

```
Initial SQL> Select VehicleName, LookQualifier,
LookValue, DamageQualifier, DamageValue
From VehicleKnowledgebase
Where LookValue= "clean" AND DamageValue=
"Null";
```

Results after fuzzy relevance computation:

Vehicle Name	Look Qual.	Look Value	Damage Qualifier	Damage Value	Simil. Value
1993 Porsche 911 Carrera 2 Cab Turbo	Null	Clean	Null	Null	1.00
1989 Porsche 911 Carrera	very	Clean	Null	Null	1.00
1977 Harley-Davidson FX	very	Clean	Null	Null	1.00

Vehicle Name	Look Qual.	Look Value	Damage Qualifier	Damage Value	Simil. Value
1999 Porsche 911 996 Carrera	very	Clean	Null	Null	1.00
1996 Porsche 993 911 Carrera	very	Clean	Null	Null	1.00
1985 Porsche 911 Turbo Carrera	super	Clean	Null	Null	1.00

Query 10: List vehicles with very light damage and mileage less than 65,000.

```
Initial SQL> Select VehicleName,
DamageQualifier, DamageValue, MileageQualifier,
MileageValue
From VehicleKnowledgebase
Where damageValue= "light" AND MileageValue <
50000;
```

Results after fuzzy relevance computation:

Vehicle Name	Damage Qualifier	Damage Value	Mileage Qual.	Mileage Value	Simil. Value
2000 Porsche Boxster	Very	Light	Null	62000	1.00
2000 Ducati 900 SS	Null	Light	Null	4100	0.50

8.4 Evaluation of the Query Processor

It is difficult to provide a performance analysis of the query-processing module, since no benchmark set of queries are available, spanning across all domains, for judging the performance of such a system. Since the concept descriptions are finally stored in a database, the system can obviously retrieve all exact matches correctly. When it comes to judging the relevance of answers to fuzzy query, the quality of retrieval is dependent on the similarity computation procedure. For example, it can be seen from the examples cited above that in some cases, the fuzzy Min-Max function seems to be too restrictive, though we have chosen it since this provides a standard way of interpreting AND and OR of entity descriptors. The similarity between two fuzzy qualifiers also depends on the expert’s judgment and the order of the qualifiers in the corresponding qualifier sets. We refrain from giving any relevance figure for this module, since acceptability of an answer generated is largely dependent on the user’s perspective.

5. Conclusion and future work

In this work, we have presented an ontology-based text-mining system, which also does enrichment of the underlying domain ontology with new and/ or imprecise concept descriptions extracted from texts and also stores

the extracted information in a structured knowledge base. The structured knowledge base is used to answer user queries related to domain entities. Storing information components on a database allows the system to collate and compare information collected from multiple sources. The use of an ontology enables information collection to be focused. The entire collection is viewed as describing a set of domain entities using common attributes. This facility is inherently provided by an ontology-based text processing system, which enables guided information extraction. We have used a fuzzy ontology structure which has been created from the initial domain ontology structure. In this extended structure, a property value is also accompanied by a value qualifier to associate a degree of precision to the value. This helps in computing relevance of a document concept to a user query concept. The degree of relevance reflects the degree of similarity between the two and generates answers even when there are no exact matches in the document repository.

Query processing is a two-step process. Initially an SQL query is constructed from the user-given query to extract entities, which have exact matches for property values stated by the user. Thereafter, the system employs fuzzy reasoning to find degree of similarity between user given concepts and text concepts. Thus imprecise matches for user query are not adjudged at par with exact matches. This aspect of query processing is totally different from the searches conducted by standard search engine.

Ontology based text processing has been mainly employed for technical domains like Biology, Zoology etc. Our emphasis has been to apply ontology based text information extraction for general-purpose user interest areas. However, since expert-created domain ontologies may not exist for all possible domains, so we have targeted our system to serve dual purpose. Starting with a seed fuzzy ontology structure, it starts analyzing domain text documents. As new information components are extracted from these, the system provides a facility to enrich the existing ontology with newly learned values and qualifiers, within the existing fuzzy ontology structure. We have worked on a number of different domains and the results obtained are quite interesting. Thus this system can be easily adapted to build text-processing systems for organizations with specific focus.

Presently, the concept extraction process uses a common framework for handling all value-qualifier sets. However, experiments suggest that the qualifier list, if treated as a partially ordered list rather than a completely ordered list may provide for more intelligent reasoning. A complete hedge-algebra based qualifier ranking along the lines of [24] may be used to compute concept relevance. Different fuzzy membership computation functions can be associated to the fuzzy reasoner to provide different kinds of reasoning facilities. We are currently working on techniques to generate the requisite grammar for a domain

through statistical analysis of documents from that domain. Integration of fuzzy relations with membership values based on reliability of source and frequency of domain concepts are also being experimented with. The system is also being extended to extract predicates from domain documents rather than simple entities. Our aim is to extend logical reasoning for these predicates through the use of extended Description Logics.

References

- [1] A. Rajaraman and P. Norvig, Virtual Database Technology: Transforming the Internet into Database, IEEE Internet Computing 2(4), July-August 1998, pp. 55-58.
- [2] Ah-Hwee Tan, Text Mining: The State of the Art and the Challenges, in: Proceedings of the PAKDD'99 Workshop on Knowledge Discovery from Advanced Databases (KDAD'99), Beijing, April 1999, pp. 71-76.
- [3] B. Adelberg, NoDoSE – A tool for Semi-Automatically Extracting Structured and Semistructured Data from Text Documents, in: Proceedings of the ACM SIGMOD Int'l Conference on Management of Data, 1998, pp. 283-294.
- [4] D. Fensel, F. van Harmelen, I. Horrocks, D. L. McGuinness and P. F. Patel-Schneider, OIL: An ontology Infrastructure for the Semantic Web, IEEE Intelligent Systems, 16 (2), 2001, pp. 38-45.
- [5] D. H. Widyantoro and J. Yen, A Fuzzy Ontology-based Abstract Search Engine and its User Studies, in: Proceedings of the 10th IEEE Int'l Conference on Fuzzy Systems, Melbourne, Australia, 2001, pp. 1291-1294.
- [6] D. W. Embley, D. M. Campbell, Y. S. Jiang, S. W. Liddle, D. W. Lonsdale, Y. -K. Ng and R. D. Smith, Conceptual-Model-Based Data Extraction from Multiple-Record Web Pages, Data and Knowledge Engineering 31(3), 1999, pp. 227-251.
- [7] G. D. Zhou and J. Su, Named Entity Recognition using an HMM-based Chunk Tagger, In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02), 2002, pp. 473-480.
- [8] H. M. Muller, E. E. Kenny and P. W. Strenber, Textpresso: An Ontology-Based Information Retrieval and Extraction System for Biological Literature. PloS Biology 2(11):e309, 2004. URL: <http://www.plosbiology.org>
- [9] H. Snoussi, L. Magnin and J. -Y. Nie, Toward an Ontology-based Web Data Extraction, in: Proceedings of the 15th Canadian Conference on

- Artificial Intelligence (AI'02). Calgary, Alberta, Canada, May 26, 2002.
- [10] J. Broekstra, M. Klein, S. Decker, D. Fensel, F. van Harmelen and I. Horrocks, Enabling Knowledge Representation on the Web by Extending RDF Schema, in: Proceedings of the 10th Int'l World Wide Web Conference, Hong Kong, 2001, pp. 467-478.
- [11] L. Crow and N. Shadbolt, Extracting Focused Knowledge from the Semantic Web, Int'l Journal of Human-Computer Studies 54(1), January 2001, pp. 155-184.
- [12] L. Delcambre, D. Maier, R. Reddy and L. Anderson, Structured Maps: Modeling Explicit Semantics over a Universe of Information, Int'l Journal on Digital Libraries 1, 1997, pp. 20-35.
- [13] M. Abulaish and L. Dey, Using Part-of-speech Patterns and Domain Ontology to Mine Imprecise Concepts from Text Documents, in: Proceedings of 6th Int'l Conference on Information Integration and Web based Applications & Services, Indonesia, 2004, pp. 91-100.
- [14] M. Shamsfard, and A. A. Barforoush, Learning Ontologies from Natural Language Texts, Int'l Journal of Human-Computer Studies 60(1), 2004, pp. 17-63.
- [15] M. Wallace and Y. Avrithis, Fuzzy Relational Knowledge Representation and Context in the Service of Semantic Information Retrieval, in: Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Budapest, Hungary, 2004.
- [16] N. Ashish and C. Knoblock, Wrapper Generation for Semi-Structured Internet Sources, ACM SIGMOD Record 26(4), December 1997, pp. 8-15.
- [17] N. Guarino, M. Carrara and P. Giaretta, Ontologies and Knowledge Bases: Towards a Terminological Clarification, in: N. Mars (Ed.), Towards Very Large Knowledge Bases, Knowledge Building and Knowledge Sharing, IOS Press, Amsterdam, 1995, pp. 25-32.
- [18] N. Kushmerick, D. Weld and R. Doorenbos, Wrapper induction for Information Extraction, in: Proceedings of the Int'l Joint Conference on Artificial Intelligence (IJCAI), 1997, pp. 729-737.
- [19] P.G. Baker, C.A. Goble, S. Bechhofer, N.W. Paton, R. Stevens, and A. Brass. An Ontology for Bioinformatics Applications, Bioinformatics 15(6), 1999, pp. 510-520.
- [20] P. Velardi, M. Missikoff and R. Basili, Identification of Relevant Terms to Support the Construction of Domain Ontologies, in: Proceedings of ACL Conference on Human Language Technology and Knowledge Management, Toulouse, France, 2001.
- [21] P. Velardi, P. Fabriani, and M. Missikoff, Using Text Processing Techniques to Automatically Enrich a Domain Ontology, in: Proceedings of ACM Conference on Formal Ontologies and Information Systems (FOIS'01), Ogunquit, Maine, 2001, pp. 270-284.
- [22] R. Stevens, C. A. Goble, I. Horrocks and S. Bechhofer, Building a Bioinformatics Ontology Using OIL. IEEE Transactions on Information Technology in Biomedicine 6(2), 2002, pp. 135-141.
- [23] S. Flank, A Layered Approach to NLP-based Information Retrieval, in: Proceedings of the 17th Int'l Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics (COLING-ACL '98), Montreal, 1998, pp. 397-403.
- [24] S. Hölldobler, T. D. Khang and H. P. Störr, A Fuzzy Description Logic with Hedges as Concept Modifiers, in: Proceedings of the 3rd Int'l Conference on Intelligent Technologies and 3rd Vietnam-Japan Symposium on Fuzzy Systems and Applications, Hanoi, Vietnam, 2002, pp. 25-34.
- [25] S. Luke, L. Spector, D. Rager and J. Hendler, Ontology-based Web Agents, in: Proceedings of 1st Int'l Conference on Autonomous Agents, 1997, pp. 59-66.
- [26] S. W. Liddle, K. Hewett and D. W. Embley, An Integrated Ontology Development Environment for Data Extraction. In: Proceedings of 2nd International Conference on Information Systems Technology and its Applications (ISTA'03), 2003, pp. 21-33.
- [27] T. Andreasen, P. A. Jensen, J. F. Nilsson, P. Paggio, B. S. Pedersen and H. E. Thomsen, Content-based Text Querying with Ontological Descriptors, Data & Knowledge Engineering 48(2), 2004, pp. 199-219.
- [28] T. Berners-Lee, J. Hendler and O. Lassila, The Semantic Web, Scientific American, May 2001, pp. 28-31.
- [29] T. T. Quan, S. C. Hui and T. H. Cao, FOGA: A Fuzzy Ontology Generation Framework for Scholarly Semantic Web, in: Proceedings of the Knowledge Discovery and Ontologies Workshop (KDO'04), Pisa, Italy, 2004.
- [30] U. Hahn and K. G. Marko, An Integrated Dual Learner for Grammars and Ontologies, Data & Knowledge Engineering 42(3), 2002, pp. 273-291.
- [31] Un Y. Nahm and R. M. Mooney, Text Mining with Information Extraction, in: AAAI 2002 Spring Symposium on Mining Answers from Texts and Knowledge Bases, Stanford, CA, 2002.

- [32] Y. Li and N. Zhong, Web Mining Model and its Applications for Information Gathering, Knowledge Based Systems 17, 2004, pp. 207-217.