ELSEVIER

Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE @ DIRECT®
Int. J. Human-Computer Studies 60 (2004) 17–63

International Journal of
Human-Computer
Studies

www.elsevier.com/locate/ijhcs

Learning ontologies from natural language texts

Mehrnoush Shamsfard*, Ahmad Abdollahzadeh Barforoush

Intelligent Systems Laboratory, Computer Engineering Department, Amir Kabir University of Technology, Hafez ave., Tehran 15, Iran

Received 10 July 2002; received in revised form 24 May 2003; accepted 15 August 2003

Abstract

Research on ontology is becoming increasingly widespread in the computer science community. The major problems in building ontologies are the bottleneck of knowledge acquisition and time-consuming construction of various ontologies for various domains/ applications. Meanwhile moving toward automation of ontology construction is a solution.

We proposed an automatic ontology building approach. In this approach, the system starts from a small ontology kernel and constructs the ontology through text understanding automatically. The kernel contains the primitive concepts, relations and operators to build an ontology. The features of our proposed model are being domain/application independent, building ontologies upon a small primary kernel, learning words, concepts, taxonomic and non-taxonomic relations and axioms and applying a symbolic, hybrid ontology learning approach consisting of logical, linguistic based, template driven and semantic analysis methods.

Hasti is an ongoing project to implement and test the automatic ontology building approach. It extracts lexical and ontological knowledge from Persian (Farsi) texts.

In this paper, at first, we will describe some ontology engineering problems, which motivated our approach. In the next sections, after a brief description of Hasti, its features and its architecture, we will discuss its components in detail. In each part, the learning algorithms will be described. Then some experimental results will be discussed and at last, we will have an overview of related works and will introduce a general framework to compare ontology learning systems and will compare Hasti with related works according to the framework.

*Corresponding author. Tel.: +98-21-6419411; fax: +98-21-6495521.
E-mail addresses: shams@pnu.ac.ir (M. Shamsfard), ahmad@ce.aku.ac.ir (A.A. Barforoush).

1071-5819/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ijhcs.2003.08.001

1. Introduction

Research on ontology is becoming increasingly widespread in the computer science community. Ontologies are used in wide range of fields such as semantic web, search engines, e-commerce, natural language processing, knowledge engineering, information extraction and retrieval, multi-agent systems, qualitative modeling of physical systems, database design, geographic information science and digital libraries.

The main problems, which are addressed in development and use of ontologies and have motivated our approach, are as follows (Barforoush and Shamsfard, 1999).

1.1. Lacking of standards to integrate or reuse existing ontologies

In recent years, there have been some efforts to create standards for ontologies by some organizations such as IEEE working group for creating Standard Upper Ontology (SUO) http://suo.ieee.org and Knowledge Systems Laboratory (KSL) at Stanford University http://www.ksl.stanford.edu/. Standardization may be done in three layers: methodology layer, language layer and content layer. In methodology layer researchers aim at developing methods and methodologies to build, integrate, share and reuse of ontologies (FernÁndez et al., 1997; Noy and Musen, 2000). In language layer the aim is to make standards on ontological structures or on representation formalisms (such as Knowledge Interchange Format (KIF) by Genesereth and Fikes, 1992 and DAML + OIL by Connolly et al., 2001) and in the content layer the goal is to create a standard for ontology contents especially the upper levels (SUMO http://suo.ieee.org/SUMO, 2002; IFF http://suo.ieee.org/IFF, 2002; Guarino et al., 1994). Although standardization, especially for methodology and language, may eliminate some integration problems in share and reuse of ontologies, there is a general agreement that content standardization of the entire world knowledge is impossible at present. So the content standardization is limited to higher levels such as the IFF core ontology by Kent (2002). Even for higher levels, according to volume and variation of essential general and domain knowledge for various domains, applications, believes, view points, cultures, etc., it would be very hard to integrate existing ontologies and converge to a standard,[1] which covers all theories and does not cause some amounts of unusable works (for intemperate definitions). Meanwhile automatic ontology learning starting from a minimal kernel, using standard methodologies and formalisms can not only fill the gap caused by the lack of content standards, but also as a complement, may facilitate the standardization process. By iterative changing the kernel and applying the learning mechanisms and evaluating the performance of system, it will be possible to standardize a minimal, domain-independent kernel. On the other hand, ontological knowledge extracted by the learning system can be used in standardization of content for the supposed environment.

[1] This is the approach, which is now used in SUO group.

1.2. Using fixed categories based on a single viewpoint

The world can be conceptualized from several different perspectives. Even for micro-worlds, which are suitable for specific domains, we may have various views of concepts and relations and so various ontologies. These ontologies have many things in common but differ from each other. A system may contain all or some of these viewpoints at the same time and use any one on demand. So having a fixed single ontology, even for a single domain, restricts us to a single viewpoint with fixed selected concepts, relations and categories. The restriction is more noticeable when we use ontologies in multiple or more general domains. A fixed hierarchy would be too rigid to accommodate widely divergent perspectives of our world (Sowa, 1995).

The ability of changing the ontology, according to the changes of the environment or according to perception of new knowledge, eliminates this problem. In dynamic ontologies, as the ontology is developed gradually, various events (such as creation, changing or updating) on ontology concepts, relations and axioms (which we call ontology elements hereafter) may occur. These changes may be because of the changes of the environment (real world, user, domain, application, time, discourse, etc.) or because of perceiving new knowledge. In both cases, the ontology should have a dynamic, flexible structure, which enables it to adapt itself with newly received knowledge and also with different perspectives of the world. This way the ontology can contain various viewpoints and use or change anyone on demand. It is obvious that such dynamic ontologies may be used as singular fixed ones in specific domains too.

1.3. Absence of full automatic knowledge acquisition methods

Construction of ontologies (especially general-purpose ontologies) is a time- and cost-consuming process. In most systems such as Cyc (Lenat and Guha, 1990) and Mikrokosmos (Nierenburg et al., 1996) at first, huge amounts of knowledge must be entered into ontology manually and then the system will be able to infer or acquire new knowledge using its large knowledge base. Using automatic knowledge acquisition methods and tools decreases the costs of development and share of ontologies.

We propose an automatic ontology building approach to build dynamic ontologies from scratch. In this approach, an ontology kernel will be developed manually and then different ontologies can be built upon it automatically, according to its environment (Shamsfard and Barforoush, 2000). The kernel would have essential meta knowledge for adding, moving, deleting and updating ontology elements. In this approach the only part of the ontology, which is not built by the system itself, is the kernel. Although it is possible to enter knowledge directly (manually or using semi-automatic tools), the system can acquire what it needs from given texts, by itself. This model uses dynamic categories to handle changes and exploits floating categories to handle multiple viewpoints. Our proposed approach although does not intend to solve the standardization problem, can eliminate the problems (in integration and reuse of ontologies) which motivated the need to content standards and also may be used as a complement for standardization efforts. It facilitates communication and integration of ontologies by building them upon a common kernel using common learning modules. On the other hand, making the ability to learn ontologies from scratch eliminates the need to huge initial knowledge bases which in turn have their own acquisition problems.

The advantages of this model are saving time & money in ontology building, covering wider range of knowledge (comparing to existing ones), being flexible to user/domain/application changes, handling various view points and ability to be linked to domain ontologies easily. The disadvantage of this model is that there may be probable slowing down in some queries and also in the first learning stages, which is the nature of learning from scratch.

Hasti project is a test bed to implement and test our proposed approach for ontology construction. The environment in this project is limited to natural language texts made of simple Persian (Farsi) sentences.

In the next three sections, after a brief description of Hasti, its features and its architecture, we will discuss its components in detail. In each part the learning algorithms will be described. Section 5 will show some experimental results and Section 6 is assigned to the discussions. There, we will have an overview of the related works and their features and compare Hasti with related works according to the mentioned features.

As Hasti extracts lexical and ontological knowledge from Farsi texts, we have several examples of Farsi sentences in this paper. In each case, the pronunciation of example is shown in italic style between double quotes (“) and its translation to English within parentheses. As the language of paper is English and readers may be not familiar with Farsi, so the written form of the examples in Farsi fonts are ignored.

2. The architecture of Hasti

Hasti is a system for automatic ontology construction, based on Persian (Farsi) text understanding. Its input is the natural language (Persian) texts and its output would be the extended (learned) lexicon and ontology. It uses an initial kernel, which has the essential Meta knowledge (primitive concepts and operators) to build an ontology.

As there is no natural language processing utilities or tools for Persian: no codified computational grammar, no collected morphological rules, no semantic or syntactic templates and even no general, public domain parser to analyse Persian texts and no computational lexicon for Persian words, so we had to define all of them primarily in Hasti.

Hasti’s lexicon is nearly empty initially and will grow gradually by learning new words. Its ontology is a small kernel at the beginning. It is formed of concepts, taxonomic and non-taxonomic conceptual relations and axioms, all as ontology elements (ontels). Hasti learns new ontels from texts to build its ontology upon the existing kernel. The kernel is language neutral and domain independent and the built ontology will be dynamic, flexible to changes and automatical expandable.

Hasti works in two modes: cooperative and unsupervised. In the former mode the user decides whenever several alternatives are probable, but in the later mode this is a duty of the system.

Input texts are converted to ontology and lexicon elements in the following steps.

- Morphological and syntactic analysis and extracting new word's features.
- Building sentence structures (SSTs).
- Extracting conceptual–relational knowledge (primary concepts).
- Adding primary concepts to the ontology.
- Ontology reorganization.

The architecture of Hasti is shown in Fig. 1.

In this section, we have a quick overview of the components of Hasti and then in the next section, we will discuss them in detail.

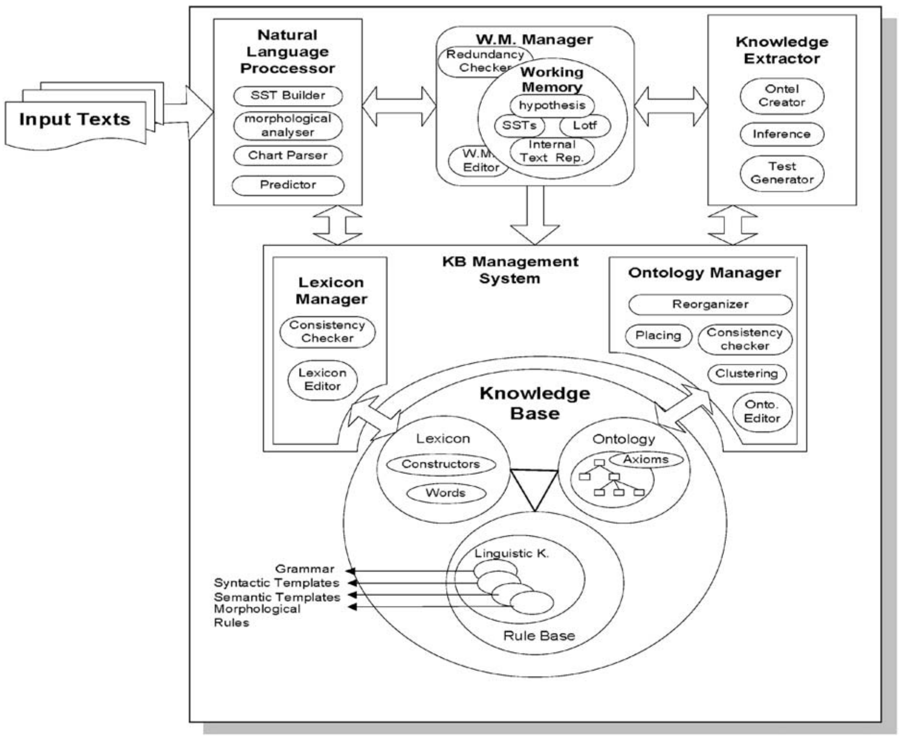

Fig. 1. The architecture of Hasti.

2.1. The natural language processor

The environment is introduced to Hasti through natural language sentences in input texts. The input sentences have to be processed and understood in order to extract their implicit knowledge. A part of these processes is done in the NLP component, which is a simple Persian text processing subsystem. It analyses sentences morpho-syntactically and writes its intermediate and final results into working memory. The NLP component is related to the KB in order to have access to the lexical, linguistic (morphological and grammatical) and ontological knowledge. It builds sentence structures to be used by the knowledge extractor and also sends lexical knowledge about words to the lexicon manager.

2.2. The working memory manager

The working memory manager manages the operations on the working memory (WM). Working memory is the place of intermediate structures to which each component of Hasti writes its intermediate results or reads the one of the other components. As a matter of fact, it is a blackboard, acting as a bridge between functional components of Hasti.

2.3. The knowledge extractor

The knowledge extractor is responsible for extracting knowledge from sentence structures. It uses semantic templates and does inferences to extract knowledge. Then the extracted knowledge about concepts and their relations will be converted to ontology elements by the ontel creator and will be written to an intermediate structure in the working memory to be placed in the ontology by the ontology manager.

2.4. The knowledge base

The knowledge base consists of the lexicon, the ontology and the rule base. Lexicon contains knowledge about words and phrases as the building blocks of language. Ontology contains knowledge about concepts as the building blocks of human conceptualization (the world model) and the rule base contains the rules and methods to extract lexical and ontological elements from texts. Each component in the knowledge base (except the rule base) is managed by its own manager. The rule base has no manager. It is fixed and is written once by the system programmer and will not be changed during the system's performance.

2.5. The lexicon manager

The duty of the lexicon manager is to manage the lexical knowledge sent to the lexicon. It adds new words and new senses of a word, updates existing words and retrieves word's knowledge to and from the lexicon. It is also responsible for keeping the lexicon consistent.

2.6. *The ontology manager*

The ontology manager is responsible for updating the ontology according to the ontels created by the knowledge extractor. This includes directing the ontel to a proper place in the hierarchy, eliminating the redundancies, keeping the consistency, updating concepts and relations in the ontology, clustering and reorganization of the ontology to obtain new concept(s) and relations regarding to common/different features, relations or attributes and values.

Next sections will describe the components of Hasti in detail. The components will be divided into two categories, the supplementary components consisting of the working memory and the knowledge base, and the functional components consisting of the NLP, the knowledge extractor and the ontology manager. The learning approach will be described in the functional components.

3. Supplementary components

Supplementary components are the working memory and the knowledge base. They are called supplementary as they store and supply the needed knowledge and information for the other parts of system. In this section, we will discuss the main structures in these components.

3.1. *Working memory*

The working memory (WM) is a blackboard to keep the internal structures of Hasti. The most important internal structures, which we refer to frequently in this paper, are text and sentence structures, indexing, hypothesis and list of primary ontels (Lopo).

3.1.1. *Text and sentence structures*

Text structure is a list of sentence structures. It contains the roles' information of the input text and keeps them by the end of a run for further referring. A sentence structure (SST) is a case frame and contains information about thematic roles, extracted from a sentence, and their relations. We divided sentences into two categories: *modal* sentences and *verbal* sentences. Modal sentences are those with copula verbs such as "*ast*"(to be), "*hast*"(to be), "*shaved*"(to become), "*shod*"(past tense of to become), "*bood*"(past tense of to be). Other sentences are verbals. The main characteristic of modal sentences is having a noun phrase as its subject and a predicate. The predicate can be a noun phrase, a prepositional phrase or an adjective phrase. The SST for a modal sentence (called a modal structure) contains three fields one for the subject, one for the predicate and one for the verb information (e.g. tense). The first two fields are list of noun phrases.

The SST for a verbal sentence (called a verbal structure) contains the thematic roles of a verbal sentence. The roles we mention in this structure are agent, patient, beneficiary, instrument, location, time, source and destination, action and action specifier. As there may be some new words in the sentence, we add a list of unknown roles to this structure too. This helps us to store any part of sentence whose role is unknown for further recognition.

3.1.2. Indexing structure

The indexing structure is used to store the ternary relation between concepts' instances, lexicon entries and input sentences. It keeps track of the right concept instance related to each word sense used in each sentence. It also shows that which sentences are evidences for each word sense and concept instance. The indexing structure is used to answer to *why*-questions about the word features and also to find a suitable word instance for a new arriving word.

3.1.3. Hypothesis

Hypothesis is a set of structures in which the different branches of system decisions are stored. The hypothesis structure is a tree and stores all possible features of a word, different SSTs for a sentence, clustering candidates, etc.

Hypothesis reduction. In the ontology learning process there are some situations in which more than one interpretation or solution exist. In such cases, various interpretations would be placed in the hypothesis structure. Each node in the hypothesis space has a certainty factor (CF), which shows how much probable it is or how much we are certain about it. Whenever an evidence confirms that a node is not valid any more, the sub-tree under that node will be deleted. In general, there should be parallel processing utilities, which allow the system testing multiple paths of the hypothesis structure in parallel, but in our system at this stage we just want to test and justify the learning algorithms. So, we simplified the hypothesis reduction and selection process in the following way.

(1) Assign a probability measure (certainty factor) to each node using some heuristics such as the ones for priority assignment (cf. Section 4.1.1) and merge set choosing (cf. the offline clustering in Section 4.3.1).
(2) Order the hypothesis space in a way that the children of each node are sorted in descending order of probability.
(3) In each case choose the first (most probable) child of the node (a depth first traverse with no backtracking!).

This method is just for testing our learning approach. An alternative method may be traversing all valid branches in parallel using threads.

3.1.4. List of primary ontels (LOPO)

Lopo is a list of primary ontels extracted from current sentence. Its structure is closed to the one of the ontology. The extracted knowledge from a sentence is gathered in this intermediate structure and after completion it will be asserted to or update the ontology.

3.2. The knowledge base

An approach for building knowledge bases in a natural language understanding system is to separate lexical knowledge from world knowledge in two different knowledge bases named as lexicon and ontology (Nierenburg and Levin, 1992). In this section we overview the lexicon, the ontology and the rule base in the knowledge base of Hasti.

3.2.1. The lexicon

The lexicon in Hasti is called Halex, which stands for Hasti Lexicon. In this section, we will discuss the features, the structure and the initial contents of Halex.

Features. Halex (Shamsfard and Barforoush, 2001) is a single language, computational lexicon. Its lexical structure is multi-structured[2] since it has two structures, one for words as the lexical units[3] (the main lexicon) and the other for word constructors like prefixes and suffixes (the constructors' structure). In other words, in Halex, lexical units are words. Sub words (affixes) are stored in constructors' structure, which is separated from the main lexicon. In Halex, entries have one level and there is no sub entry.[4] On the other hand, Halex uses a single entry for all of the senses of a word, so it has a one-to-one mapping between entries and lexical units. As different senses may differ not only in meaning (semantic or pragmatic features) but also in morphological or syntactic features, so the size and content of fields in different entries is not fixed. Halex is a dynamic lexicon, which learns new words and updates the existing words during the system's performance.

Initial contents. Halex contains a few words at the beginning. All of the permitted prepositions (Farsi equivalents for in, on, at, to, from, for, above, under), pronouns (Farsi equivalents for I, you, he/she/it, we, you, they), copula verbs (Farsi equivalents for to be and to become), conjunction sign (Farsi equivalents for and), direct object sign ("*ra*")[5] and modifier sign ("*e*")[6] are lexemes, which are listed in the lexicon at its birth time. Some other words from verbs, nouns, adjectives and adverbs are also chosen from input texts randomly and are added to the lexicon to test the system. Other words will be learnt by the system.

Structure. Halex is a set of entries in which each entry consists of a word and a set of word senses. Each word sense has four major lists about morphological, syntactic, semantic and pragmatic knowledge. The information stored for each word sense in Halex is as follows.

[2] *Lexical structure* refers to if there is a unified (single) or multiple structure(s) for lexicon entries.

[3] *Lexical Units*' are the constituent units of the lexicon, which may be words, sub-words (affixes) or super-words (complexes).

[4] An entry refers to all information about a lexical unit and in Halex its depth is one.

[5] Direct object sign has no equivalent in English.

[6] Modifier sign sometimes is equal to "*S*", or "*of*" and sometimes has no equivalent as well.

- *Morphological knowledge*: The main portion of the morphological knowledge in Hasti is encoded in morphological rules and the constructors' structure. The other portion, which is related to words, such as single/plural form of a noun, the word's root or the past tense of a verb, is appeared in the lexicon. In other words, we mostly mention the inflectional information rather than derivational ones.
- *Syntactic knowledge*: This part contains the grammatical category and subcategories of the word. A word category shows that the word is a noun, verb, adjective, adverb, determiner or preposition. The subcategories contain more details about the words and are different for each category. For example, we have binary features such as single/plural, general/specific, countable/uncountable … for nouns and transitive/intransitive and past/present for verbs. We also add some syntagmatic information such as permissible complements (prepositional phrase) for a verb here.
- *Semantic knowledge*: In Hasti, the meaning of a word is determined by its pointers to the ontology. Each word is related to one or more concepts in the ontology. The corresponding concept(s) shows the main meaning of the word. Special cases, features and selectional restrictions for the word will be stored as its semantic knowledge in the lexicon. So the main portion of what other lexicons (e.g.: Wordnet (Miller, 1995)) call semantic knowledge such as paradigmatic information can be inferred from the ontology.
- *Pragmatic knowledge*: The pragmatic knowledge is designed to be located in the ontology and be accessible from the lexicon. It could contain the information about the word's context or environment. The pragmatic field in the lexicon should keep some pointers to partitions of ontology, which show different contexts in which the word occurs and has different pragmatic interpretations. Currently it contains a list of evidence sentences in which the word is seen.

Various senses of a word may differ in morphological, syntactic, semantic or pragmatic features. Fig. 2 shows the structure of an entry in Halex.

In this figure the *i*th word of Halex is shown which has *N* different senses. As the figure shows each word may have several senses for each we store morphological, syntactic, semantic and pragmatic knowledge. For each field, we have a certainty factor (CF), which shows how much we are sure about the corresponding information. For CFs, 255 means that the information is defined by the system engineer (initiated at the beginning), values between 129 and 254 mean that the information is confirmed by the user number (CF-128) explicitly, 128 means that the information is confirmed with more than 127 evidences (confirming sentences in text) and values less than 127 mean that the information is confirmed by CF number of sentences in the text (evidences will be stored in the *indexing structure*). Total CF will be calculated by a function of detail CFs. This function is simply the addition now.

Constructors. Besides the main lexicon for words, there is a structure to store affixes named as constructors. Constructors are morphemes, which make words, in combination with other morphemes, words and word roots. We store constructors in the constructors' structure, which is a part of lexicon. In Hasti, constructors are limited to suffixes and prefixes. Each entry in this structure has three fields, the affix

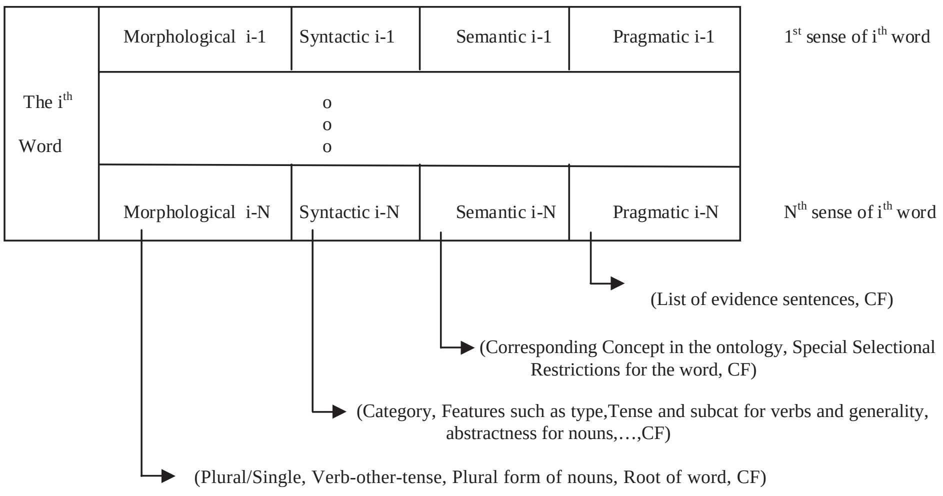
The ith Word
Morphological i-1
Syntactic i-1
Semantic i-1
Pragmatic i-1
1st sense of ith word
o
o
o
Morphological i-N
Syntactic i-N
Semantic i-N
Pragmatic i-N
Nth sense of ith word
(List of evidence sentences, CF)
(Corresponding Concept in the ontology, Special Selectional Restrictions for the word, CF)
(Category, Features such as type,Tense and subcat for verbs and generality, abstractness for nouns,…,CF)
(Plural/Single, Verb-other-tense, Plural form of nouns, Root of word, CF)

Fig. 2. The structure of a Halex entry.

type, base which contains features of the lexeme that the affix combines with and result which contains the features of the result of combination. The affix, itself, is the key to access to this structure. For example, the suffix "*ban*" (keeper) in Persian attaches to the end of names and build names with a meaning of protection such as "*darban*" (doorkeeper), "*darvazeban*" (gatekeeper) or "*baghban*" (gardener or garden keeper). For "*ban*" the affix type is suffix, the grammatical category of base and result are both noun and the result's semantic feature points to the concept of protection in the ontology.

3.2.2. The ontology

In this section, we will discuss the features, the structure and the initial contents of the ontology in Hasti.

Definition 1. The ontology is defined by $O = (C, R, A, \mathrm{Top})$ in which C is the set of all concepts (including relation concepts), R is the set of all assertions in which two or more concepts are related to each other, A is the set of axioms and Top is the most top level in the hierarchy. R itself is partitioned to two subsets, H and N. H is the set of all assertions in which the relation is a taxonomic relation and N is the set of all assertions in which the relation is a non-taxonomic relation.

Features. The basic features of the ontology are as follows.

- The ontology is built upon an initial kernel which
  - Contains primitive elements (concepts and operators) and basic meta knowledge for construction, correction and modification of the ontology.
  - Is language neutral and domain independent.
- The built ontology will be dynamic and flexible to changes.

- The ontology may contain various viewpoints and theories at once and use any one on demand by implying floating relations and floating categories.[7]
- The formalism used to represent the ontology is a simplified version of KIF.
- The ontology contains concepts, taxonomic and non-taxonomic relations[8] and axioms.
- The relations may be n-ary ($n > 1$).

Initial contents. The ontology consists of concepts, relations and axioms. Concepts are organized in a network whose some of edges make an inheritance hierarchy. In other words, the ontology presents not only the "isa" hierarchy of concepts but also any other relations between them. There are some axioms about the concepts and relations too. These axioms help the system to understand and use the ontology elements better.

At the first step, the content of the ontology is limited to the kernel. It grows in a middle-out manner around the kernel during the system's performance. The growth direction is called middle-out since the ontology grows incrementally and received ontels have no specific order.

The top level of the kernel consists of linguistic-motivated concepts. As the ontology is learned from natural language texts, we chose the primary concepts as *object*, *action* and *property* (corresponding to nouns, adjectives and verbs). As the initial knowledge base is very small and we just know or predict the syntactic category of words, so the chosen primary concepts help us to pre-locate the concepts in the ontology and then place them in their appropriate location by receiving additional knowledge about them. There are also other concepts in the kernel, which are used to relate concepts or define compound concepts. Some of the basic top-level concepts (classes) of the kernel are as follows.

- Top
  - Object
  - Action
  - Property
  - Set
  - Relation[9]
    - Hierarchical: sub-class, instance-of
    - Has-prop
    - Has
    - Thematic-roles: agent, patient, instrument, beneficiary, …
    - =
    - Spatial-rel
    - Temporal-rel: Before, After, …
    - Member-of

[7] In each viewpoint, there are some relations which may be invalid or changed in other viewpoints. Switching between these is a feature of our proposed approach. This is carried out by weightening the ontology elements in various viewpoints and will be discussed later.

[8] Although relations are concepts too, but we name them separately for better understanding.

[9] Each <relation> has a correspondence potential relation (shown by P-<relation>). A concept c1 has a Potential relation with another concept c2, if there exists an instance of c1, which occurs in the relation with c2 or an instance of it.

Table 1
Examples of initial axioms

```
(<=> (is-transitive ?r)                              (<=> (isa ?x ?y)
     (and (sub-class ?r Relation)                         (or (instance-of ?x ?y)
          (forall (?x ?y) (=> (and (?r ?x ?y)                 (sub-class ?x ?y)
                                   (?r ?y ?z))                (and (isa ?x ?z)
               (?r ?x ?z)))))                                 (isa ?z ?y))))
(is-transitive sub-class)                            (forall (?x) (isa ?x top))
                                                     (<=> (disjoint ?x ?y)

(<=> (is-transitive-over ?r1 ?r2)                        (forall (?c) (=> (isa ?c ?x)
     (and (sub-class ?r1 Relation)                            (not (isa ?c ?y)))))
          (sub-class ?r2 Relation)                   (=> (antonym-props ?x ?y)
          (forall (?x ?y) (=> (and (?r1 ?x ?y)            (and (isa ?x property)
                                   (?r2 ?y ?z))               (isa ?y property))
               (?r1 ?x ?z)))))                                (forall (?c) (=> (has-prop ?c ?x)
(is-transitive-over has sub-class)                                 (not (has-prop ?c ?y)))))
```

And some initially inserted axioms are shown in Table 1.

Structure. The topology of the ontology is a multiple directed graph with different edge types, which includes the following.

- An *isa*[10] hierarchy in the form of an acyclic directed graph containing the concept taxonomies in which a node may have more than one parent.
- A directed graph of concepts denoting non-taxonomic conceptual relations.
- Logically represented axioms.

The structure of the ontology in Hasti is simple and uniform. The inheritance hierarchy, concept frames, relations and axioms are all represented in kif-styled lists. The first element of each list is a relation and the rest of it denotes the related terms in the form of:

(<relation> {<related-concept>}*)

Some examples are as follows.

(a) (sub-class human object)
(instance-of sara-1 human)
show a two level isa-hierarchy in which sara-1 is a human and human is an object.

(b) (has sara-1 book-1) shows a possession relation between sara and her book.

(c) (has-prop book-1 color red) shows that an attribute for this book is 'color' which its value for the book is red (the book's color is red).

[10] As the ontology contains both concept level and instance level, by *isa* we mean both the sub-class and instance-of relations throughout the paper.

```
(d) (=> (subclass ?s ?c)
        (forall (?x)
        (=> (instance-of ?x ?s)
            (instance-of ?x ?c)
     )))
```

means that if s is a subclass of c then every instance of s is an instance of c too.

3.2.3. The rule base

The rule base contains the rules and methods to extract lexical and ontological elements from texts. It consists of linguistic rules, inference rules and heuristics. It is fixed and is written once by the system programmer and will not be changed during the system's performance.

Linguistic rules. The language grammar, morphological rules, syntactic and semantic templates are built-in linguistic knowledge in Hasti.

- The *grammar* chosen for initial input sentences covers a subset of Persian sentences. As Hasti learns from scratch and its lexicon has few entries at the beginning, the grammar is limited for initial states in which the percentage of unknown words in a sentence is very high (this decision is made to reduce the branching factor of the hypothesis space). The grammar can be improved and cover more complex structures gradually while system learns more words and gathers more knowledge. In other words, the grammar, would be simple at initial states because of the lack of initial knowledge and can become complicated incrementally. This is a natural solution just like what occurs about young children. They can understand simple sentences at the beginnings and their ability improves while they learn more. In our experiments, we used simple texts, made for primary school students to start the learning process. It is obvious that the syntactical simplicity of texts is due to the NLP part (to build SSTs) and is not related to other components of Hasti.
- *Morphological rules* indicate how to make or analyse plural nouns, and various forms of verbs. They also show how to use the constructors' structure to make or analyse inflectional and derivational words.
- *Syntactic templates* are the functional attachments to the grammar, which help the system to make SSTs from sentences. They indicate the thematic roles of parts of speech. For example, in a sentence with a transitive active verb the noun phrase before "*ra*" (direct object sign) is the direct object or patient of the sentence. The syntactic template for such sentences is shown below:

  $$S \rightarrow \text{np} \ll \textit{agent} \gg \text{vp1}$$

  $$\text{Vp1} \rightarrow \ldots|\text{np} \ll \textit{patient} \gg \text{ra trans-verb} \ll \textit{action} \gg | \ldots$$

  To indicate the thematic roles, the system uses some subcategorization information coded at verb entries in the lexicon. If the entry does not denote the role of an NP after (or before) a specific preposition then the system assigns the default (or possible) role(s) for the complement according to its preposition by using the syntactic templates.

- *Semantic templates* are used to convert SSTs to ontels by extracting taxonomic and non-taxonomic relations and axioms. They denote the semantic relations between different roles in a sentence. The proposed semantic templates (Shamsfard, 2003) are divided into four categories.

(a) Copular templates, to discover hyponymy, meronymy and attribute-value relations from copular sentences.
(b) Verbal templates, to extract thematic (case) roles and relate their concepts to the concept of the verb, in non-copular sentences.
(c) Noun phrase templates, to discover relations between different parts of a noun phrase such as attribute-value, possession, etc.
(d) Axiom templates, to extract axioms from conditional or quantified sentences.

Parts of the semantic templates of copular (modal) sentences are shown in Table 2. There are some examples for each template. The examples' descriptions are shown in Table 3 and their numbers are used in the example part of Table 2.

To make the tables clear, let's look again at modal structures. A modal sentence has a noun phrase as its subject and a predicate. The predicate can be one of the followings types (in each case the predicate is underlined):

- ○ A noun phrase such as "*Josef mard e bozorg – i ast.*" (Josef is a great man.) or "*baradar e bozorg e Maryam doost e khoob e hamkar e jaded e Daniel ast*" (Maryam's old brother is the good friend of Daniel′s new colleague).
- ○ An adjective phrase like "*Rang e an sib sorkh bood.*" (The color of that apple was red) or "*lebas e boland e Sara tamiz va ziba ast.*" (Sara's long dress is *clean and nice*.).
- ○ A prepositional phrase such as "*In angoshtar az tala e sefid ast.*" (This ring is of white gold) or "*ketab e dastan e Maryam rooye mize gerd ast.*" (Maryam's story book is *on the round table*).

In each case a part of a modal structure will be filled and some conceptual information according to the above semantic templates will be extracted from this structure. For instance, as the first row of Table 2 shows, if the predicate is an adjective phrase then three cases may occur in which the main noun of subject may be a property or not and if it is a property then the adjective in the predicate may be an instance of the main noun of subject or not. In each of these cases, the table shows the extracted knowledge from the sentence. Lets look at example #1. In this sentence, the main-noun of the subject is "*rang*" (color) which is a property and the adjective in the predicate is "*sabz*" (green) which is a kind of color and so there is an *isa* relation between these two. In this case, the extracted ontel will be a non-taxonomic relation (color) between the modifier of the subject (machine) and the predicate (green). The extracted knowledge will be shown by (Has-prop machine color red).[11]

[11] It is equivalent with (color machine red).

Table 2
Part of the semantic templates for modal sentences

Grammatical state of the sentence: S → Subject predicate copula-verb, Subject → Np, Np → main-noun + adjs + modifiers				
Conditions			Conclusion	Ex. no.
Condition-1	Condition-2	Condition-3		
Predicate → Adjs	Main-noun of subject *isa* property	Adj *isa* main-noun of Subject	Main-noun of the subject is an attribute of the main-noun of its genitive modifier and the adj is the value of this attribute.	1
		Otherwise	Main-noun of the subject is an attribute of the main-noun of its genitive modifier and the adj is an adjective for this attribute.	2
	Otherwise		The adj is the value of an attribute of the main-noun of the subject.	3
Predicate → NP	Main-noun of subject *isa* property (attribute)	The predicate *isa* main-noun of subject	Main-noun of the subject is an attribute of the main-noun of its genitive modifier and the main-noun of predicate is the value of this attribute.	4
	Main-noun of the subject is synonym to the predicate's		Add the features of the predicate (adjectives and genitive modifiers) to the subject's main noun.	5
	There is an *isa* relation between subject and predicate		Make sub/super class relations between the main nouns of the subject and the predicate.	6
				7
	Otherwise		The subject is/was/becomes equal to the predicate.	8
Predicate → Pp	The prep. is a location-prep		The predicate indicates the location of the subject.	9
	…			

Table 3
Persian examples for semantic templates

No	Persian pronunciation	Translation to English	Extracted ontels
1	"*Rang e machin sabz ast.*" (color's machine green is)	Machine's color is green.	(has-prop machine-0 rang-0 sabz-0)
2	"*Rang e machin e Sara ziba ast.*" (color of machine's Sara nice is)	The color of Sara's machine is nice.	(has sara-0 machine-1) (has-prop machine-1 rang-1) (has-prop rang-1 atx-0 :ziba-0)
3	"*Sara mehraban ast.*" (Sara kind is)	Sara is kind.	(has-prop sara-0 : atx-1 mehraban-0)
4	"*Naam e in derakht naarvan ast.*" (Name of this tree narvan is)	The name of this tree is Narvan.	(has-prop derakht-1 naam naarvan)
5	"*Lebas e oo lebas e zibaee bood.*" (Dress's she dress—a-nice was)	Her dress was a nice dress. (= Her nice dress)	(has oo lebas-1) (has-prop lebas-1 atx-2 ziba-1)
6	"*Asb yek heivan ast.*" (Horse an animal is)	Horse is an animal.	(sub-class Horse Animal)
7	"*An selah yek kard bood.*" (That weapon a knife was)	That weapon was a knife.	(sub-class kard selah) (instance-of kard-0 kard)
8	"*Daniel baradar e Miriam ast.*" (Daniel brother's Miriam is)	Daniel is Miriam's brother.	((has miriam-0 baradar-0) & (= baradar-0 daniel-1) OR (is-baradar-of miriam-0 daniel-1)
9	"*Ketab rooye miz ast.*" (Book on table is)	The book is on the table.	(spatial-rel rooye ketab-1 miz-1)

Another example is #6 in which a taxonomic relation is learned. In such cases a *sub-class* relation will be held between the main noun of the subject and the main noun of the predicate.

In general, in cases which the predicate is a noun phrase and we know a few about features of the subject and predicate, a co-reference (equality) relation will be held between the heads (main nouns) of the subject and the predicate. Then new relations will be extracted according to some co-reference rules. Two main co-reference rules are as follows:

First rule: (= > (and (HAS a b) (= b c)) (IS-B-OF a c))

Second rule: (= > (and (= a b) (instance-of a c) (instance-of b c)) (merge a b))

Table 4 shows examples of applying these rules.

Verbal templates are mainly used to extract non-taxonomic relations between thematic roles in a sentence both with themselves and with the verb. The main relations between a concept of a role and the concept of verb is the role (such as agent, patient, beneficiary, …) and relations between various roles depend on the verb. For example, in a simple sentence "*Maryam sib e sorkh ra khord*" (Maryam ate the red apple) following ontels may be extracted by verbal templates:

(agent Maryam-0 eat-0)
(patient apple-0 eat-0)
(Eater apple-0 Maryam-0)
(Eats Maryam-0 apple-0)

Table 4
Examples of applying co-reference rules

Reference sentence	Initial extracted relations	New relations after applying the rule
Daniel is Miriam's brother	(has Miriam-0 brother-0) (= Daniel-0 brother-0)	(Is-brother-of Miriam-0 Daniel-0) (a new relation)
Miriam's green home is Daniel's big home	(instance-of home-0 Home) (has Miriam-0 home-0) (has-prop home-0 color green-0) (instance-of home-1 Home) (has Daniel-0 home-1) (has-prop home-1 size big-0) (= home-0 home-1)	(instance-of home-0 Home) (has Miriam-0 home-0) (has-prop home-0 color green-0) (has Daniel-0 home-0) (has-prop home-0 size big-0)

Noun phrase templates are other templates to extract hyponymy, meronymy and possession relations. They include adaptations of Hearst' patterns[12] (Hearst, 1992) for Persian, the exception template, the modification template and others to extract relations between various parts of a noun phrase (head and modifiers). As an example the exception template implies the hyponymy relations as follows.

Exception template: Phrases such as…{all|every} NP_0 except NP_1 $\{(\text{and}|,)NP_i\}^*$ … ($i > 1$), imply that for all NP_i ($i \geqslant 1$) there is a hyponym relation between NP_i and NP_0 (sub-class NP_i NP_0).

As another example, consider the above simple sentence (Maryam ate the red apple). Applying the modification template on it will result in the following ontel:

(Has-prop apple-0 atx-0 red-0) if red is unknown or (Has-prop apple-0 color red-0) if red is known as a color.

Function of axioms template is described in Section 4.2.1.

Inference rules. The knowledge extractor and the ontology manager use logical inference techniques (such as proof by resolution) and backward chaining and forward reasoning methods to extract new knowledge or check the consistency, validity or existence of a knowledge element. The procedures to handle these tasks are coded in this part of the rule base.

4. Functional components

Functional components consist of the NLP component, the knowledge extractor, the ontology manager and the lexicon manager. In this section, we will describe the structure, parts, functionality and learning methods used at the first three of these components. The lexicon manager is a database manager, which has no learning module its own and is not discussed in this section.

[12] Such as "NP{, NP}*{,} (and|or) other NP".

4.1. The natural language processing component

The natural language processing component analyses input sentences morpho-syntactically. It has five major functional components:

- Lexical analyser.
- Morphological analyser.
- Syntax analyser (chart parser).
- SST (Sentence Structure) builder.
- Predictor.

The above components use some linguistic and world knowledge to extract lexico-conceptual knowledge from Persian texts. An overall schema of the NLP component (surrounded within dotted lines) is shown in Fig. 3.

Input texts pass through the lexical analyser to be broken to sentences and words. Then the syntax analyser, which is a top-down chart parser, converts sentences to their corresponding parse trees. It will invoke the morphological analyser or the predictor on demand. In other words, during the processing, whenever an unknown word was met, the morphological analyser will be called to convert it to one of the existing words in the lexicon. If no matches found, the predictor will be activated to learn the word's features. To extract the lexical knowledge of a word, the predictor tries to find matching grammar rule(s) for that sentence. As we suppose that all input sentences are valid (grammatically correct), the fired grammar rule(s) help the

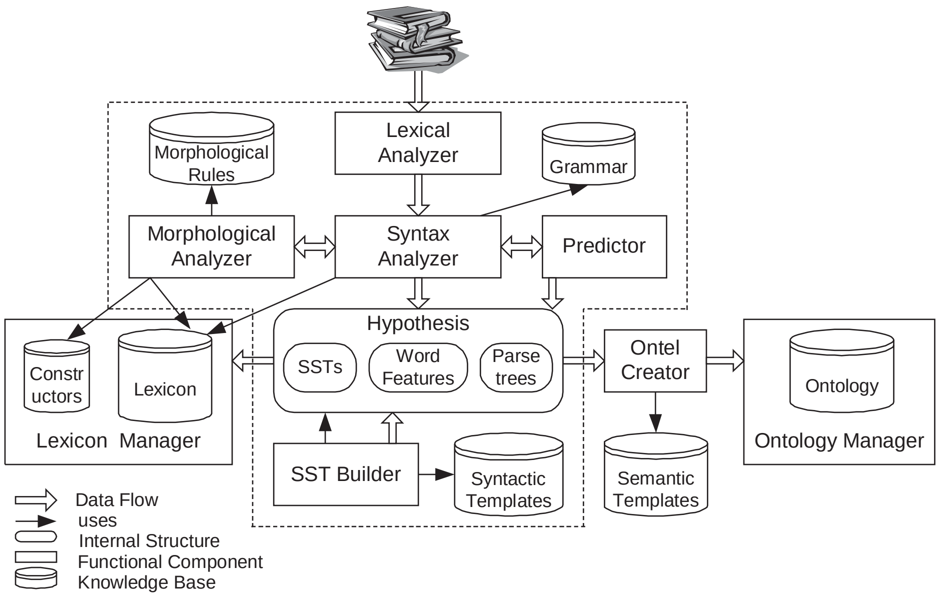

Fig. 3. The overall schema of the NLP component of Hasti.

predictor to predict the category(s) of the unknown word. New words with their probable features will be written in the hypothesis structure and then may be passed to the lexicon manager to be added to the lexicon if needed. At last the SST builder will make sentence structures (described in Section 3.1) from parse trees according to some syntactic templates (described in Section 3.2.3). SSTs indicate the thematic roles in a sentence and syntactic templates indicate how to convert syntactic roles (part of speech) to thematic roles.

At the rest of this section we will describe the process of learning lexical knowledge about words.

4.1.1. Learning lexical knowledge

At the beginning, there are few entries in the lexicon for which only a part of syntactic information (including their category) is filled. Other entries are empty and even the meaning of known words are unknown. So the lexicon acquisition task in this project consists of adding new words to the lexicon and updating or completing the information about the known words. There are two subtasks to learn new words: acquiring lexical features and discovering the meaning. The first subtask is performed in the NLP component and the second one in the knowledge extractor.

The learning method in this part is a linguistic based, template driven method, which uses some heuristics too.

Whenever the system faces an unknown word in a sentence, which cannot reduce it to a known word, tries to learn its features through morphological, syntactic, semantic and pragmatic analysis. To extract the lexical knowledge of a word, the following steps will be gone through.

- Invoke the morphological analyser to reduce the new word to a root if possible. The morphological analyser uses the constructors' structure, which contains the information about affixes. For each unknown word, according to its predicted syntactic category(s) (verb or non-verbal), a part of morphological rules will be activated. For instance, *person-number-time determination* rules analyse unknown verbs to determine their root as the main constituent and the indicated person, number and time as their features. *Prefix and postfix deletion* rules are general rules to analyse non-verbal unknown words. They test the word to be combined with known affixes listed in the constructors' structure. The order in which the affixes are tested is sometimes important. So there are some meta rules which indicate the affix test orders.

As the result of this stage, four cases may occur. The new word may be reduced to the following.

(i) A known root and known affix(es).
(ii) An unknown root and known affix(es).
(iii) A known root and unknown affix(es).
(iv) An unknown root and unknown affix(es).

In Hasti, we consider the first two cases in which the affixes are known, because we don't mention affix (constructor) learning in this phase.[13] In the first two cases, the constituents of the new word will be passed to the syntax analyser. Then the syntax analyser makes two branches in the hypothesis structure, one in which the new word is a solid new word and the other in which the new word consists of the extracted root and affix (es). The difference between the above two cases is in the probability measure of these hypothesis. There is a simplifying heuristic, which says that the branch of being new solid word is more probable in the second case and less probable[14] in the first case.

- Find out the new word's category according to the grammar. As we suppose that all input sentences are valid (correct), the fired grammar rule(s) will determine the category of the unknown word. If more than one rule fire then more than one probable category will be found for the word. In such cases, system tries to reduce the branching factor (possible solutions) as much as possible. Some of the partial solutions to reduce the branching factor are as follows.
  - There exists an alternative of being pronoun for all unknown nouns. So we add all pronouns to the lexicon to eliminate this *over-branching* problem.
  - There exists an alternative of being adjective for some unknown nouns in noun phrases. For such cases in which there are no other information to assign priorities, we used some simplifying heuristics to define priorities for grammatical roles in sentences. For instance, in the modifier position of a noun phrase being an adjective has higher priority than being a noun while in the head position it is reversed (being noun has higher priority).
  - There exists an alternative of being copula verb for some unknown intransitive verbs. So we add all copula verbs to Halex in order to reduce the branching factor for sentences with unknown verbs.

  For example in "*Sara ketab e nafis ra forookh*" (Sara sold the valuable book), when the word "*nafis*" (valuable) is unknown, according to the grammar, "*nafis*" can be a noun, a pronoun or an adjective. By adding all pronouns to the lexicon we reduce the first level branching factor from 3 to 2 ("*nafis*" may be a noun with higher probability or an adjective with lower one). Another example is "*Sara beh madreseh raf.*" (Sara went to school). In this sentence, if "*raft*" (went) is unknown then it can be either a copula verb or an intransitive verb. Adding all copula verbs eliminates the branching factor from 2 to 1 ("*raft*" is an intransitive verb).
  - Facing natural language ambiguities are some other cases in which there may be more than one parse tree for a sentence. For example in noun phrases such as "*Ketab va daftar e Sara*" (Sara's book and notebook) or "*Zan va mard e pir*" (Old man and woman) we have ambiguities.

[13] There is a plan to handle the third case in further expansions of the project to learn new constructors according to known roots.

[14] Its probability is not zero.

"(*ketab*) *va* (*daftar e Sara*)" and "(*ketab va daftar*) *e Sara*" are possible interpretations for the former sentence, and "(*zan*) *va* (*mard e pir*)" and "(*zan va mard*) *e pir*" for the later. To resolve these natural ambiguities, we again used heuristics to define priorities for operators such as "*va*"(and) and "*e*"(modifier sign) according to their context and selectional restrictions of the connecteds (if any). In cases which "*va*" has higher priority the second interpretation will be more probable.

It should be noticed that these heuristics will be applied just when there is no other alternative (other information) to decide from and none of the applied ambiguity resolution methods have their prerequisites to work.

- Add the new word to the lexicon except for:
  - Different persons of a verb (we just keep the root).
  - Regular plural form (root plus plural signs) of nouns.

  In other words each derivation, which keeps the meaning of the word (inflectionals), will be derived by morphological analysis. Others, which change the word's meaning (derivationals), will be stored in the lexicon but their semantic background will be inferred from morphological process using the constructors' structure.

The next step is to find out the new word's meaning. This will be done in the knowledge extractor component and the ontology manager will add, correct or find the corresponding ontology element(s) for the word.

4.2. The knowledge extractor

The knowledge extractor is responsible for extracting knowledge from sentence structures. It uses semantic templates and does inferences to extract the knowledge in two levels: sentence level and text level. At the sentence level, which will be performed after processing each sentence, the extracted knowledge about concepts and their relations will be converted to ontology elements (ontels) by the ontel creator. The ontel creator converts words and phrases to inter-related concepts and makes additional relations between these concepts according to their roles in the sentence structures. The resulting facts will be written to Lopo to be used by the ontology manager. At text level, knowledge will be extracted by logical inferences from acquired knowledge.

The rest of this section discusses the various parts of the knowledge extractor and the learning process, which is used to extract knowledge at both sentence and text levels. *The learning approach in the knowledge extractor is a combination of logical, template driven and semantic analysis methods.*

Ontel creator. The ontel creator is responsible for primary concept formation. It converts verbs and noun phrases to inter-related concepts and determines their original place in the ontology. Then additional relations between these concepts will be made according to their roles in the sentence structures using semantic templates. Semantic templates indicate how to convert thematic roles to conceptual relations (cf. Section 3.2.3). The created ontels will then be placed in their proper position in the ontology by the ontology manager.

Inference engine. The inference mechanism in Hasti is based on resolution. The inference engine does backward chaining to deduce new knowledge from the existing ones. It uses the inference rules which are located at the rule base and works on knowledge elements which are located at the ontology to infer new knowledge elements.

Test generator. Test generator generates tests to extract more knowledge about unknown aspects of concepts. For instance, testing the possession of a feature or testing participation in a relation in order to find paradigmatic similarities (cf. similarity measures in Section 4.3.1) between two concepts are performed by this part.

4.2.1. Knowledge extraction at sentence level

The words' meanings are shown by their pointer(s) to ontology elements. There is a single or a set of concept(s)/relation(s) correspondence to each word in the ontology. For presenting the meaning of a new word we may need to define a new concept/relation or in other words create a new ontel. This is the duty of the ontel creator. Following are the steps to create an ontel.

(1) Create a primary concept (hereafter called p-concept) for each unknown word meaning. In this stage the system should find or build the proper concept for the new word. It also finds or creates an instance of that concept (class) to assign the current sentence to, in the instance level. In the process of instance assignment, *default reasoning* is used. It moves backwards (chronologically), starting from the most recent instance built for this word sense, until the first match (no contradiction) is reached. Contradiction is defined by having disjoint attributes, values or relations.

(2) Attach new p-concepts under their original fathers. The original father of a p-concept may be extracted in the following ways:
   (a) Extracting from current sentence: This will be done by hyponymy-extraction semantic templates such as Hearst's templates (Hearst, 1992), or our proposed templates like the exception template and modal templates in which the predicate is an NP (Shamsfard, 2003). For example in "*Hame e parandegan joz panguan*" (all birds except penguin) the p-concept created for 'bird' will be the original father of the p-concept created for 'penguin' and in "*John yek ensan ast*" (John is a human) the p-concept created for 'human' would be the original father of the p-concept created for 'John' in the hierarchy.
   (b) Inferring from the corresponding word's syntactic category: In cases which the sentence does not indicate an original father for the concept (like the case for 'bird' in the first example and for 'human' in the second one), we attach it to the highest level of ontology under the *kernel* according to its syntactic category and grammatical role (e.g.: nouns under *object*, adjectives under *property* and verbs under *action*).

(3) Extract primary conceptual relations between the created p-concepts according to the SST, using semantic templates. Various kinds of semantic templates and some examples of them were discussed in Section 3.2.3. There are also some examples of extracting conceptual relations from modal sentences shown in Table 3. For verbal sentences the main conceptual relations are thematic (case) roles. For example in sentence: "*John ab nooshid*" (John drank water) the corresponding p-concept for 'John' is a potential-agent for the corresponding p-concept of 'drinking' and the p-concept for 'water' is its potential patient. This means that 'water' relates to the class of drinkables and 'John' to class of drinkers. Other relations may be also extracted by using reasoning and inferring from the internal text representation using the world knowledge.

(4) Learn Axioms. The SSTs of conditional sentences or compound sentences with a quantifier phrase (in which the subordinate clause is about the quantified NP) will be converted to axioms in the following steps.
   (a) Separate the antecedent and consequent parts of the sentence. In conditional sentences this is a simple task. In compound sentences, which we mention to create axioms from, the antecedent is the subordinate clause and the consequent is the base sentence.
   (b) Complete the incomplete and necessary roles of each part according to the other part using heuristics for resolving referential and omission ambiguities.
   (c) Create ontels for each part separately and name them antecedent ontels and consequent ontels respectively.
   (d) Make an implication relation ($\Rightarrow$) between the antecedent ontels and the consequent ontels.
   (e) Convert the unbound instances to variables in the axiom.
   (f) Extract and add facts to the axiom to denote the implicit features of the implication such as the temporal or spatial relations and causality between antecedent and consequent of the axiom.

Steps (a) and (b) will be done by the NLP component and steps (c)–(f) by the knowledge extractor.

As a simple example, to convert the sentence "*Agar ensani sam bokhorad, mimirad*" (If a human eats poison, will die) or similarly the sentence "*Har ensani sam bokhorad, mimirad*" (Any human who eats poison will die) to an axiom, at first the NLP component will separate the sub-sentences 'A human eats poison' and 'will die' as the antecedent and the consequent. The SSTs built for these sub-sentences are:

(and (instance-of ?x Human)(instance-of poison-0 Poison)
(instance-of eat-0 Eat)(Agent ?x eat-0)
(Patient poison-0 eat-0)) as the antecedent ontel and
(instance-of die-0 Die) as the consequent ontel.

Then using the *NP-ommision* heuristic the system will find out that the deleted subject of the consequent is the same as the subject of the antecedent. So the consequent ontel will become

(and (instance-of die-0 Die)(Agent ?x die-0)).

Next step is to create an axiom by making an implication relation between antecedent and consequent ontels. The axiom will be

```
(⇒ (and  (instance-of ?x Human)(instance-of poison-0 Poison)
         (instance-of eat-0 Eat)(Agent ?x eat-0)(Patient poison-0 eat-0)
         (and (instance-of die-0 Die) (Theme ?x die-0)))
```

Then the unbound instances and classes should be converted to variables. Unbound instances and classes are those that are not assigned to proper nouns or words with specific references. At this step the instances poison-0, eat-0 and die-0 will be converted to variables ?poison-0, ?eat-0 and ?die-0 (or to ?y, ?z, ?w if we denote in the setup), respectively. Now it's time to insert implicit facts to denote causality and temporal relations between the antecedent and the consequent. According to the sentence and using the conditional templates, we infer that if the antecedent occurs then the consequent will occur after it so we assign a time stamp to each part of axiom and relate them by an *after* relation. At last the above sentence will be converted to the following axiom.

```
(⇒  (and  (instance-of ?x Human)(instance-of ?poison-0 Poison)
          (instance-of ?eat-0 Eat)(instance-of ?t1 Time)
          (Agent ?x ?eat-0)(Patient ?poison-0 ?eat-0)(Time ?eat-0 ?t1))
    (and  (instance-of ?die-0 Die)(Theme ?x ?die-0)(instance-of ?t2 Time)
          (Time ?die-0 ?t2)(After ?t2 ?t1)))
```

In some cases we can convert these sentences to assertions about concepts and their relations too, but we convert them to axioms in order to use them in logical inferences and not use them in clustering.

After creating ontels from current sentence, it's time to locate them in their appropriate place in the ontology. This will be done by the ontology manager using an incremental (online) clustering technique.

4.2.2. Knowledge extraction at text level

Knowledge extraction at text level refers to extracting the knowledge, which cannot be inferred from single sentences, such as implicit knowledge of text, causes and effects, pre and post conditions of verbs, etc. At the current stage, part of the implicit knowledge of text is extracted by logical reasoning at the inference engine. Inheritance laws, transitive rules and other axioms, are used to extract this part of knowledge.

Generating tests to extract knowledge would be also performed at this level. For instance, to extract or prove paradigmatic similarities between two concepts, we shall assume that both have same features or occur in same relations. If this assumption was not explicitly mentioned in the text or could not be inferred from text the test generator generates some tests to evaluate it. The feature (relation) which the test is performed for is called the *desired feature* (*relation*), the concept which is being tested is the *testing concept*, the concept which the comparison is made with, is called the *base concept* and the sentences in which the base concept has the desired feature (relation) are *reference sentences*. To do the test, the test generator uses the indexing structure to find reference sentences in which the base concept has the desired feature or relation and then replaces the base concept with the testing concept in the reference sentences and asks user to validate them. If the sentences are confirmed by the user then the testing concept has the desired feature or relation too and so is similar to the base concept. As an example, suppose that we want to find the similarity between concepts of apple and orange, and there have been some sentences in the text denoting that apple is edible (for example Miriam ate the apple). So the testing concept is orange, the base concept is apple and the reference sentence is (Miriam ate the apple). By doing the substitution the test would be (Miriam ate the orange) which is also meaningful and semantically correct. So apple and orange are paradigmatically similar (cf. Section 4.3.1) since both may be eaten by Miriam.

4.3. The ontology manager

The ontology manager is responsible to update the ontology according to the ontels. This includes directing the ontel to a proper place in the hierarchy, eliminating the redundancies and conflicts, updating concepts and relations in the KB according to new facts, merging/splitting existing concepts (classes) to obtain new concept(s) regarding to common/different features, attribute and values or relations. *The learning approach in the ontology manager is based on semantic analysis, which uses heuristics too and the main learning task in the ontology manager is conceptual clustering.* Our clustering method relies on the assumption that concepts occurring in the same relations with similar concepts can form a common super concept.

4.3.1. Clustering

Hasti does both hierarchical and non-hierarchical clustering since the built clusters have both hierarchical and non-hierarchical relations with one another. It learns multiple hierarchies by a *multi clustering* technique from *multiple points of view*. “Multi clustering” means that each concept may be clustered to more than one cluster or in other words, in the built directed graph each node may have many fathers and/or many children. It also may have many non-taxonomic relations to other nodes. By “multiple points of view” we mean that the process of finding similarities may be controlled by assigning appropriate weights to relations and related terms according to the user’s believes or the domain model (we will discuss this matter later).

On the other hand, Hasti learns and clusters new concepts in both on-line and offline modes. The on-line mode does incremental clustering and the offline mode does periodic clustering.

On-line or incremental clustering. The on-line mode works at sentence level. In this mode receiving a concept with some information about, extracted from current sentence, invokes a placing module to find the concept's father in the appropriate level of abstraction. The concept will be transferred from its original place (under its original father) toward its best-matching place in the hierarchy according to the new arrived knowledge about it. For new concepts, the original father would be a node determined by the ontel creator (cf. Section 4.2.1). The *place-it* module moves the p-concept from its original place downward (toward the hierarchy leaves) in order to find its *most specific father* (*Msf*). The most specific father (*Msf*) of a concept is the one that (1) matches the p-concept features, (2) there is not a single match in its children, and (3) has the lowest level among all candidates. Finding a match is a complicated procedure. The simplest form of match is the absence of antilogy (contradiction) and a simple form (the most restricted form) of it is having no features more than the p-concept's. In other words the *Msf* of a concept is the deepest descendant of its original father whose feature set is a subset of the feature set of the p-concept. To make the point clear let's look at the following definitions:

Definition 2. The feature set of a concept (c) in an ontology (O) (shown by $F(c)$) is a set containing all assertions in which the concept (c) occurs. In other words it denotes all of the relations, which the concept (c) has with other ontels.

$$F(c) = \{A \mid (c \in A) \wedge (A \in O)\}.$$

Definition 3. The non-taxonomic feature set of a concept (c) (shown by $Nf(c)$ is a set of assertions in which c has a non-taxonomic relation with any other concept. It is the intersection of $F(c)$ and N.

$$Nf(c) = \{A \mid A \in (F(c) \cap N)\}.$$

Definition 4. The Inherited non-taxonomic feature set of a concept (c) (shown by $Inf(c)$) is the union of the non-taxonomic feature set of c and the non-taxonomic feature sets of the ancestors of c from its immediate father(s) to the root.

$$Inf(c) = Nf(c) \cup \bigcup_{i=father(c)}^{Root} Nf(i).$$

Definition 5. The most specific father (*Msf*) of a concept (shown by $Msf(c)$) is the deepest descendant of its original father whose feature set is a subset of the feature set of the concept and the cardinality of its feature set is the maximum among other candidates. To formulate this definition, suppose that $Orf(x)$ returns the original father of x, *Descendants* (x) returns all of the descendants of x from its immediate children to leaves and $Card(x)$ returns the cardinality (number of members) of x, then we will have:

$$\begin{aligned}
&M = Msf(c) \Leftrightarrow \\
&\quad ((M \in (Descendants(Orf(c))) \wedge (Nf(M) \subseteq Nf(c)) \wedge \\
&\quad (\forall x;\ x \in Descendants(M) \Rightarrow (Nf(x) \not\subset Nf(c))) \wedge (Card(Inf(M)) is \text{ max})).
\end{aligned}$$

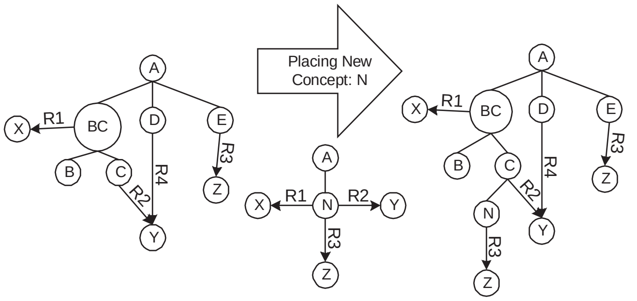

Fig. 4. Placing a new concept in the hierarchy.

As the definitions show, in moving downward from the original father to find the Msf, if none of the children of a father match the p-concept, then the transfer operation will be stopped and the p-concept will be attached under the father. If more than one child match the p-concept then using a look-ahead technique, system will mark the current father and examine lower levels to find matches. If still there exists more than one choice then the p-concept will be attached under the one, which has the most common features with p-concept. Otherwise, if the choices are similar according to the number of common features then the p-concept will be attached under the marked father.

To find a match, each attribute has a marker, which shows if its value can be merged, added, rewritten or changed by the new value. There is a match between non-similar attributes if (1) they are not antonyms, (2) their values are not antonyms, and (3) their values can be matched according to their markers. Antilogy recognition will be possible with respect to the introduced features of the attribute's concept in the ontology.

For example, suppose the initial graph is the one in the left-hand side of Fig. 4 and the newly arrived p-concept is *N*. The right-hand side of Fig. 4 shows the graph after placing *N*. As it has been described, to find the *Msf* of *N* we start from its original father which is *A*, then moving downward, there are three candidates: $\{BC, D, E\}$. *Nf* (*BC*) and *Nf* (*E*) both are subsets of *Nf* (*N*). So the candidates will be reduced to $\{BC, E\}$. If we continue the test for all children of each node in the candidate list, we can find *C* as the next candidate since *Nf* (*C*) is a subset of *Nf* (*N*). So we replace the *BC* with its child (*C*) in the candidate list and the candidate list will become $\{CE\}$. As the nodes in the candidate list have no child to continue so the downward movement will be stopped and now it is time to choose one among remained candidates. As $(Card(NF(C) \cap Nf(N)) = 2)$ and is greater than $(Card(NF(E) \cap Nf(N)) = 1)$, so the node *C* will be chosen as the *Msf* of *N*.

Offline or non-incremental clustering. After formation of p-concepts and placing them in the hierarchy, they may be merged based on common features or split based on separation factors in order to create new concepts and/or new relations or cluster similar concepts. Two concepts can be merged to a more general concept or be placed under a common super concept, if (1) the weight of their common features become more than the user-defined *merge threshold*, (2) the cost of their differences be less than the user-defined *split threshold*, and (3) there is no antilogy between their features. The merge and split functions are performed in the offline mode of clustering. The offline mode does periodic clustering, after each P sentences, which P may be determined manually or automatically.[15]

The main criterion for being in a cluster is in having the same relations with the same ontels. In other words, concepts with same relations to same ontels are candidates to be merged or put under a common super concept. We developed and applied two algorithms to cluster the concepts, a concept-based algorithm and a relation-based algorithm and introduced three[16] levels to apply our algorithms:

(1) *First level*: At this level having the same direct relations to the same concepts is the merge criterion. For example if we have (Has-prop Apple Color Red) which means that there is a color relation between apple and red, and (Has-prop Machine Color Red) which means that there is a color relation between machine and red, then apple and machine both has the same relation (color) with same concept (red) and so are similar.
(2) *Second level*: At this level the criterion is having the same indirect relations to the same concepts. By indirect relations we mean relations, which can be indirectly derived by logical inference or hierarchical inheritance. To derive these relations we use axioms such as transitivity and inheritance rules like *if x has y and y isa z then x has z* or *if x is-role-i-for y and y isa z then x is-role-i-for z* and so on. For example if we have

    (P-agent Human Breathe-1), (Sub-class Breathe-1 Breathe-2),
    (P-instrument Breathe-1 Lung-1)
    (P-agent Fish Breathe-0), (Sub-class Breathe-0 Breathe-2),
    (P-instrument Breathe-0 Gill-1)

    Then by applying the above rules (transitivity of p-agent over sub-class) we will have:

    (P-agent Human Breathe-2) and (P-agent Fish Breathe-2)

    and so human and fish are similar since both are P-agents of Breathe-2 which means both breathe.
(3) *Third level*: At this level the criterion is having same relations to similar concepts. Similar concepts are those which have became similar in the previous levels.

The concept-based algorithm. In this algorithm, as Table 5a shows, we build an upper triangular similarity matrix (simatrix) in which the similarity measure for each pair of concepts is computed (by the calculate-similarity function). Then the pair with maximum similarity, whose similarity is also greater than the merge-threshold,

[15] To simplify the system at this stage, we determine the P manually, but there is a plan to automate this task in future. In automated version P should be small at the beginning and should be increased as more concepts are learned.

[16] Of course, there are more than three levels of similarity for example having similar relations (instead of same relations) to similar concepts is the other one. At this stage, we just mention the above three.

Table 5

(a) The concept-based algorithm

```
Begin /*Concept-based clustering*/
  Do
    For all concept pairs $(c, c')$ in the ontology do
      Simatrix $(c, c')$ ← calculate-similarity $(c, c')$
    Candidates ← Choose-pair (simatrix)
    Merge-pair ← find-best (candidates)
    Merge (merge-pair)
  While no more pairs remain to merge
End /*Concept-based clustering*/
```

(b) The relation-based algorithm

```
Begin /*Relation-based clustering*/
  For all r in Antecedents (RELATION) do
    $Nfr$ ← null

    /*set $Nfr$ to the subset of $Nf(r)$ in which r is the first element*/
    For all a in $Nf(r)$ do
      If first$(a)$ = $r$ then
        $Nfr$ ← $Nfr + a$
    candids ← null
    For $i=1$ to valence $(r)$ /* number of arguments*/ do
      candids ← candids + order-relation $(Nfr, i)$
  Sort-Candidates /*in descendent order of inter-group similarities*/
  For all merge-set in candids do
    Merge (merge-set)
    Reorganize
End /*Relation-based clustering*/
```

Table 6

(a) The algorithm for merging classes

```
Begin /*Merge $(c_1, c_2, \ldots, c_n)$*/
  $Mscf$ ← Most-specific-common-father $(c_1, c_2, \ldots, c_n)$
  $Cf$ ← create-super-concept $(c_1, c_2, \ldots, c_n)$
  commons ← common features of $c_1, c_2, \ldots, c_n$
  Add commons to the feature set of $Cf$
  For $i=1$ to $n$ do
    Remove commons from the feature set of $c_i$
    Delete any direct isa relation between $c_i$ and $Mscf$
    Make an isa relation between $c_i$ and $Cf$
  Make an isa relation between $Cf$ and $Mscf$
End /*Merge $(c_1, c_2, \ldots, c_n)$*/
```

(b) The algorithm for finding the *Mscf*

```
Begin /*Most-specific-common-father $(c_1, c_2, \ldots, c_n)$*/
  Common-fathers ← fathers $(c_1)$

  For $i = 2$ to $n$ do
    $fc[i]$ ← fathers $(c_i)$

    Common-fathers ← Common-fathers $\cap$ $fc[i]$
  For all $f$ in common-fathers do
    $D(f)$ ← 0

    For $i=1$ to $n$ do

      $D(f)$ ← $D(f)$ + distance $(f, c_i)$
  Return the $f$ with minimum $D(f)$
End /*Most-specific-common-father $(c_1, c_2, \ldots, c_n)$*/
```

will be chosen (by the choose-pair function) to form a new super concept. If more than one pair has the maximum similarity then there are some heuristics such as the first pair, the pair with maximum common parents or the pair with at least one middle-concept to choose one among others (coded at find-best function). Then the selected pair will be passed to the merge function (described in Table 6) to be merged. In this approach, each intermediate (non-leaf and non-kernel) node in the primary conceptual hierarchy has at most two children, but the hierarchy is not a binary tree as each node may have more than one father.

The relation-based algorithm. In this algorithm (shown in Table 5b), we consider just the non-taxonomic relations instead of all concepts as the central role in clustering. For each non-taxonomic relation in the ontology, we look for concepts related by that relation. If at least one argument (related concept) is common between assertions about that relation, then the set of other arguments (called merge-set) will be good candidates to be merged. After checking all relations in the ontology, a list of several merge-sets will be built. Then the merge set will be chosen which has the maximum weight and has the maximum similarity between its members. If we suppose that all weights (which are described later) are equal then the longest merge-set in this list will be chosen to be merged. The order-relation function in the algorithm, gets *Nfr* and *i* as inputs and returns a list of merge-sets. Each merge-set consists of *j*th (for $j \neq i$ and $j \neq 1$) arguments of assertions in *Nfr* whose *i*th arguments are the same. *Nfr* is the subset of $Nf(r)$ in which *r* is in the relation position (which is usually 1).

Although the concept-based algorithm is flexible to changes in similarity measures or rules and its code is simpler, but its performance depends heavily on candidate choosing heuristics. In a simple choose-first heuristic, there may be some bad-classifications due to the order of pair choosing. On the other hand the experiments showed that the cost (time and memory) of the concept-based algorithm is more than the relation-based one. The similarity matrix is usually large and sparse and we spend a long time to compare similarity between unrelated concepts (concept pairs with no similarities). So we chose the relation-based algorithm to do the clustering task.

Similarity measure. In clustering algorithms there is a similarity measure indicating the similarity between two classes. In the literature, two main different types of similarity have been addressed (EAGLES, 1996).

- "Semantic similarity" (the so-called paradigmatic or substitutional similarity) and
- "Semantic relatedness" (the so-called syntagmatic similarity).

Two words can be considered to be semantically similar if they may be substituted for one another in a particular context. For example, in the context I read the book, the word book can be replaced by magazine with no violation of the semantic well-formedness of the sentence, and therefore the two words 'book' and 'magazine' can be said to be paradigmatically similar.

Two words can be considered to be semantically related if they typically or significantly co-occur within the same context. For instance, cut and knife are syntagmatically similar since they typically co-occur within the same context.

Hasti mostly considers semantic similarity to cluster concepts. The similarity measure is computed as follows:

$$Sim(a,b) = \sum_{j=1}^{MaxLevel} \left(\sum_{i=1}^{Card(cm)} \left(W_{\mathrm{cm}(i).\mathrm{r}} + \sum_{k=1}^{Valence(cm(i).r)} W_{\mathrm{cm}(i).\mathrm{arg}(k)} \right) \right) \times L_j;$$

$$\mathrm{cm} = Nf(a) \cap Nf(b).$$

As the formula shows, to find the similarity between two concepts c and c', we shall first find their common features. Common features are the ones in the intersection of non-taxonomic feature sets of c and c'. Then for each common feature the weights of the relation and the related terms will be accumulated. The accumulated weights will then be added together in each similarity level. The similarity measure is the sum of the production of accumulated results in level-constants (l_j). The level constants are constants assigned to each similarity level, which decrease by the level increase. It means that lower similarity levels are more significant than upper ones. The reason is obvious as by moving upward in similarity levels we move from the most similarity (at the first level) to the less similarity (at the third level).

Weighted relations and related terms. As it has been described before, the built ontology is dynamic and flexible to changes of user/domain and can handle various viewpoints. To develop this flexibility we assigned some weights to relations and related terms. The weights show the significance and importance of the weighted element (a concept, a relation or a combination of both) in the domain or in the user's point of view. A combined element may be a feature for a concept (a set of the feature and the concept) or a relation along with its related concepts (a set of the relation and the related terms). For example, in a general domain to be animating or ability to think are major features for human while to be tall or having a pet are not (however in a special domain, being tall or having a pet may be an important feature and so having a high weight). So it is important to distinguish the distinctive properties (major features) for a class/concept from others and also distinguish the significant concepts in a domain or in a user's belief or in a viewpoint. This will be done by weightening.

Clustering direction. The hierarchical clustering can be done in two directions: top-down and bottom-up. In the top-down direction at first all concepts are collected in a single cluster. Then the cluster will be splitted and most similar concepts make sub-clusters. In other words the hierarchy will be built by incremental specification from top to bottom. In the bottom-up direction at first there are N clusters for N concepts (there is one concept in each cluster). Then the most similar clusters will be merged and make super-clusters. In other words the hierarchy will be built by incremental generalization from bottom to top. Hasti has modules to do clustering in both directions and use them on demand.

In bottom-up clustering, similar concepts will be merged to make common super concepts. Similar concepts will be found by one of the two algorithms to find clustering candidates. After merging n concepts, the resulted concept will be placed under *the most specific common father* (*Mscf*) of original concepts.

Definition 6. Suppose $P(c)$ is the set of all parents (fathers and ancestors) of c and $HDist(a,b)$ is the hierarchical distance of a and b which is the minimum number of "isa" edges between a and b. Then the Mscf of n concepts $(c_1, c_2, \ldots, c_n)$, which will be shown by $Mscf(c_1, c_2, \ldots, c_n)$ will be calculated as follows:

$$C = Mscf(c_1, c_2, \ldots, c_n) \Leftrightarrow \left(\left(c \in \bigcap_{i=1}^{n} P(c_i)\right) \wedge \left(\sum_{i=1}^{n} Dist(C, c_i) \text{ } is \text{ min}\right)\right).$$

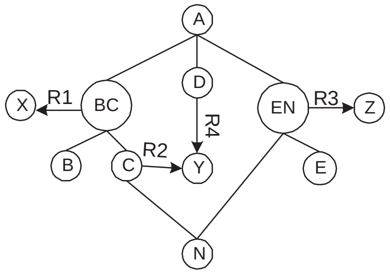

Fig. 5. Clustering the graph of Fig. 4.

As the formula shows the *Mscf* of *n* concepts, is their common father with lowest level in the hierarchy or the node whose sum of distances to original concepts is minimum. The algorithms for merging classes and finding the *Mscf* are shown in Table 6a and b.

In top-down clustering, a concept whose majority of children has a common feature may be splitted to two sub concepts having and not having that feature. Before splitting, in the cooperative mode the system asks user if the other children have that feature as well. The question will be generated by test-generator component in the knowledge extractor. If the answer was positive then the common feature would be transferred to the common father otherwise the splitting will be performed.

In the unsupervised mode in which we do not have the user intervention, system uses the closed world assumption. In other words, if there is no assertion denoting that the children have that feature then they don't have. So the splitting will be performed. In these cases the reasoning will be non-monotonic. If a piece of knowledge denoting that the child has the feature arrives later, then the ontology would be corrected by adding the new knowledge and re-clustering.

Fig. 5 shows the clustering of the graph obtained in Fig. 4 by the bottom-up, relation-based algorithm with unique weights.

4.3.2. Ontology refinement and reorganization

The ontology should be refined and reorganized periodically in order to eliminate redundancies, remove superfluities and unnecessary parts, prune the graph and obtaining new ontels. The period may be as soon as for each sentence or as late as after processing the entire text or in a cycle between the two margins. Reorganization consists of clustering (merging similar concepts or splitting a concept to coherent sub-concepts), transferring common features upward in the hierarchy, pruning unnecessary concepts and branches and eliminating some redundancies.

The refinement task will be done by the following four main modules:

Factoring. Common features between all children of a node will be transferred to the node to avoid redundancies. In other words, if all children of a node have a feature we suppose that their common father would have that feature and the children would inherit it from their father. This is a case of inductive learning in

Hasti.

$$\forall x;\ x \in \bigcap_{c \in Children(p)} Nf(c) \Rightarrow (\forall c;\ c \in Children(p) \Rightarrow (Nf(c) \leftarrow Nf(c) - \{x\})) \\ \wedge (Nf(p) \leftarrow Nf(p) \cup \{x\}).$$

Pruning. An unnecessary node, which is not referred to by any lexical unit and its own feature set is empty, should be deleted and its children should be transferred to under its immediate father.

$$\forall x;\ ((x \in mid - concepts) \wedge (Card\ (father\ (x)) = 1) \wedge (Nf(x) = \Phi)) \Rightarrow \\ ((children\ (father\ (x)) \leftarrow (children\ (father\ (x)) \cup children\ (x)) \wedge delete\ (x)).$$

Merging branches. If the set of children of a node is a subset of children of the node's sibling the whole sub-tree of the node and its children will be transferred to under its sibling and its children will be deleted from the set of its sibling's immediate children.

$$\forall x, y;\ (sibling(x, y) \wedge (children(x) \subseteq children(y)) \Rightarrow ((children(y) \leftarrow children(y) \\ - children(x) + x) \wedge (children(Mscf(x, y) \leftarrow (children(Mscf(x, y) - x))).$$

Discarding redundant paths.[17] If a node has a direct hierarchical relation to one of its ancestors (non-immediate fathers such as grandfathers) then the relation should be deleted.

$$\forall x, y, z;\ (father(x, y) \wedge Isa(y, z) \wedge \exists w;\ assertion(w) \wedge ((w = \text{“}(sub - class(x, z)\text{”}) \\ \vee (w = \text{“}(inst - of(x, z)\text{”}) \Rightarrow delete(w).$$

As an example, a simple scenario of ontology management, which is a sequence of clustering, receiving new knowledge, clustering and pruning is shown in Fig. 6.

It is noticeable that reorganization of an ontology does not change its semantic and conceptual features, it rather simplifies its usage and completes the clustering. So we can infer anything from an ontology after reorganization, which we could infer before it. In reorganization we delete relations or concepts which will never be used or can be obtained from others and create new useful relations or clusters.

4.4. An example

Evaluation of our system was based on small-scaled test cases in wide range of domains: texts from the Persian book of first grade of primary school, from children's storybooks and from technical reports. Results showed that the proposed

[17] In knowledge-based systems sometimes we intentionally make redundant knowledge and keep them. In other words, sometimes, we keep redundancies to improve the efficiency of system. On the other hand, too many redundancies may slow down the system because of enlarging the search space. So the system engineer should find an optimal amount of redundancies, which makes the best performance. In our case, as at this stage, we do not mention the best performance, we simply remove redundant paths and keep any other redundancies, which are explicitly mentioned in input texts.

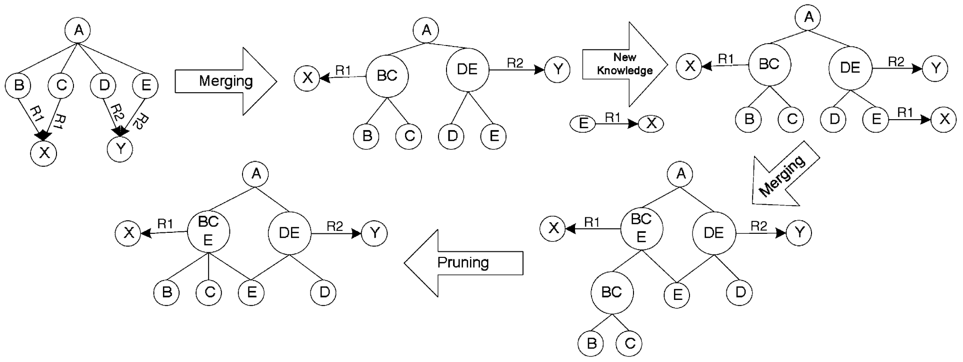

Fig. 6. A simple scenario of ontology management.

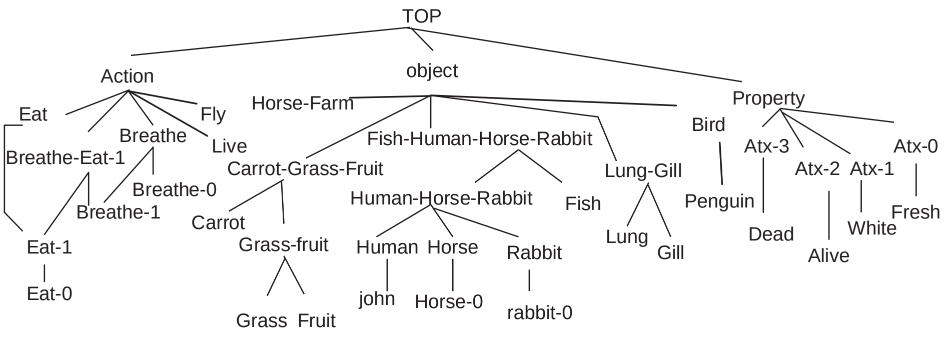

Fig. 7. The resulted hierarchy after processing the text in Table 7.

Table 7
A simple text to show the ontology learning process

Fish live in water and breathe by Gill but Humans live on land and breathe by lung. Horses and rabbits breathe by lung too. Blacky is John's horse. Blacky eats fresh grass. John is a human. He has a farm and eats fresh fruits in it. Robby lives in that farm. Robby is the name of a white rabbit who eats carrot. … All birds except penguin fly. …_… Any human who is killed, is not alive. If a human is not alive, he is dead. …

approach could be applied for building both general purpose and domain specific ontologies. In the following, we will show the results of processing a sample text.

4.4.1. A simple test

To show the process of learning lexical and ontological knowledge, we have chosen a small piece of text consisting of 11 sentences containing 43 words which 27 of them are unknown (have no entry in the lexicon). The ontology is limited to the kernel (primitive concepts). Table 7 shows the translation of the simple text to English. The underlined words are unknown and have no corresponding meaning(s)

Table 8
Part of non-taxonomic knowledge and axioms extracted from the sample text in Table 7

```
(P-Agent Human-Horse-Rabbit Breathe-eat-1)
(P-Agent Fish-Human-Horse-Rabbit Breathe)
(P-Agent Fish Breathe-0) (P-Agent Bird Fly)
(Not (P-Agent Penguin Fly))
(P-Agent Human-Horse-Rabbit Eat)
(P-Instrument Lung-Gill Breathe)
(P-Instrument Lung Breathe-1)
(P-Instrument Gill Breathe-0)
(P-Agent Human-Rabbit Eat-0)
(P-Patient Carrot-Grass-Fruit Eat-1)
(P-Patient Grass-Fruit Eat-0)
(P-Has-Prop Grass-Fruit Atx-1 Fresh)
(= Horse-0 Blacky) (Is-horse-of Blacky John)

(Has john blacky) (P-Has Human Horse)
(Has-Prop Rabbit -0 Atx-0 White)
(P-Has-Prop Rabbit Atx-0 White)
(Name Rabbit-0 Robby)
(P-Has Rabbit Name) (Has John Horse-Farm)
(P-Loc Farm Eat-Live)
(=> ( and (instance-of ?x Human)
            (instance-of ?y Kill)
            (Patient ?x ?y) )
   (Not (Has-prop ?x Atx-2 alive)))
(=> ( and (instance-of ?x Human)
       (Not (Has-prop ?x Atx-2 alive)))
   (Has-prop ?x Atx-3 dead))
```

in the ontology. The weights of all relations are simply zero for primitives and one for others. The merge and the split thresholds are both set to 2.

After processing the text, placing p-concepts, clustering and reorganization of the ontology, the result will be obtained. Part of the resulted ontology is shown in Fig. 7 and Table 8. Fig. 7 shows part of the taxonomic relations (the hierarchy) and Table 8 shows part of the non-taxonomic knowledge and the axioms extracted from the sample text. There are also 27 new words learned at the lexicon which their syntactic (grammatical category) and semantic (corresponding concept in the ontology) fields are completed.

The results then will be presented to user to correct or rename concepts if needed. In each case the concept name and its feature set will be presented to user. For example the concept Human–Horse–Rabbit has three sub-concepts (children): Human, Horse and Rabbit and itself is a child of Fish–Human–Horse–Rabbit. Instances of this class can eat (P-agents of the concept of Eat) and breathe by lung (P-agents of Breathe-1 whose p-instrument is the concept of Lung). After providing the above information to user, he may rename it for example to Land-livings.

5. Experimental results

Evaluation of our system was based on test cases in different domains and complexities, from children's books to technical reports. Tests were repeated for various values of the following system parameters.

- Weights of concepts and relations: Changing weights resulted in different clustering results and so different ontologies, which all were acceptable according to the viewpoint presented by weights.
- Clustering algorithm: Testing the two proposed clustering algorithms (concept- and relation-based) showed that the relation-based algorithm is more efficient according to time and memory.

- Parameters of placing task: In on-line clustering, we defined two parameters: the comparison operator and the place operator. As it was discussed, each new concept to be placed in the ontology, should move from its original father downward to under (the place operator) a node whose feature set is a subset (comparison operator) of the feature set of the concept. These operators were selected after testing various operators (e.g. as a sibling instead of *under* or a superset or having most commonalities instead of *subset*) and showed the best outputs in evaluations.
- Generality and domain: The tests were carried out on general and domain specific texts. Results showed that our approach is suitable for building both general and domain ontologies.
- The amount of prior knowledge: We tested our method on various situations of pre-knowledge. The learning methods worked on empty ontology and lexicon as well as non-empty ones. The experiments showed that the emptiness of the initial lexicon and ontology does not affect our learning algorithms; it rather eliminates the bottleneck of knowledge acquisition for building the initial lexicon and the base ontology.

5.1. Evaluation

For evaluation, we separate the entire process into two phases: (1) converting sentences to sentence structures (SSTs) and (2) building ontologies from SSTs. To evaluate each phase, we assumed that its inputs are correct. At each phase, we applied the methods on various pieces of general and specific texts and computed the precision and recall measures. Precision shows the number of extracted correct results divided by number of total extracted results while recall shows the number of extracted correct results divided by the number of total corrects. To evaluate the results, the outputs of the system are compared with the results built by independent people, experts and non-experts.

5.1.1. Phase one: converting sentences to sentence structures

While building correct sentence structures depends on correct recognition of morphological and syntactic features of words and appropriate application of syntactic templates, so the evaluation of the first phase (converting sentences to SSTs) is itself divided into three sub-phases: morphological analysis, syntax analysis and building SST.

(a) *Morphological analysis*: The average precision for analysis of each word knowing that it is a verb or a non-verb is 82% and average precision of morphological analysis of texts with 200–300 unknown words is about 73.4%. The recall for the morphological analysis phase is 100%, as the system records all possible results. The results are calculated with respect to the prior knowledge of the system about affixes. It means that we have not included the errors caused by the lack of initial knowledge about unknown affixes in this part.

(b) *Syntax analysis and creation of Parse trees*: The most error prone part of this phase is determining the word's syntactic category by the predictor. The average precision increases by decreasing the percentage of unknown words in sentences (Fig. 8). The average precision of building a Parse tree for a sentence is 66% and the average recall for this task is 76.3%.

(c) Building SSTs: To build SSTs we use syntactic templates which determine the default roles for parts of speech. On the other hand, the system knows default sub-categorizations of each preposition. In evaluating Hasti, we tested the system in both modes of using and not using default values. In the latter mode in which the system does not use default values, it considers all possible roles for a phrase according to its position in the sentence. As Fig. 9 shows, using default values results in increasing the average precision with the cost of decreasing the average recall. It also increases the performance by decreasing the volume of needed computations.

According to our test cases the average precision for determining thematic roles and building SSTs by using default values is 62.5% and without using default values is 47.5%. Thus, the introduced syntactic templates increase the precision.

Table 9 shows the precision and recall values in the first phase of evaluation.

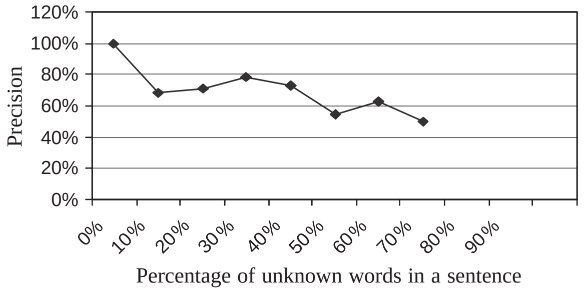

Fig. 8. The precision of building Parse trees based on the percentage of unknown words in a sentence.

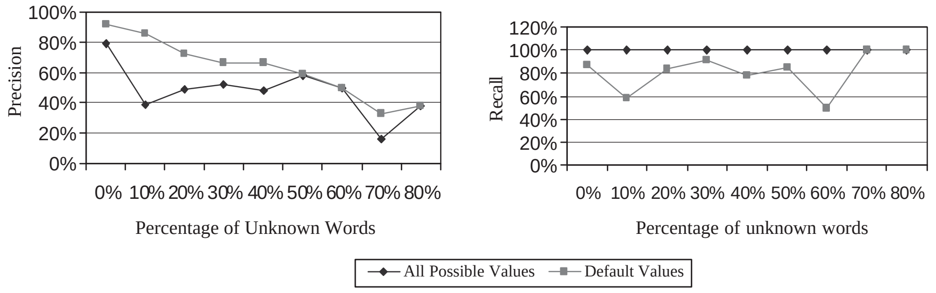

Fig. 9. Precision and recall for building SSTs from Parse trees.

Table 9
Summary of the evaluation results for the first phase (building SSTs)

Criteria	Morphological analysis (%)	Syntax analysis of building Parse trees (%)	Determining thematic roles and building SSTs	
			Using default values (%)	All possible values (%)
Precision	82	66	62.5	47.5
Recall	100	76.2	85.14	100

Table 10
Summary of the evaluation results for the second phase (ontology building from SSTs) for two groups of samples

Criteria	Texts from first grade of primary school (%)	Texts in computer domain (%)
Unknown words	86	81
Precision	97	78
Recall	88.5	79.6

5.1.2. Phase two: extracting ontological structures from SSTs

At this phase precision and recall will be calculated for extracting Ontels consisting of concepts, taxonomic and non taxonomic relations. This phase is evaluated at sentence level. Table 10 shows the evaluation results for two types of texts: general texts (selected from the Persian book of first grade of primary school) to extract general ontologies and technical texts (in computer domain) to extract domain ontologies.[18] In this experiment, the ontology kernel had merely seven concepts and seven conceptual relations.

6. Discussion

Ontologies are widely used in information systems, and ontology construction has been addressed in several research activities. In recent years, two approaches have been concerned to solve the ontology engineering problems.

(1) Development of methods, methodologies, tools and algorithms to *map*, *merge* and *alignment* of existing ontologies (Noy and Musen, 2000; Ryutaro et al., 2001; Lacher and Groh, 2001).

(2) Development of methods, methodologies, tools and algorithms to *acquire* and *learn* ontologies (semi) automatically (to name a few, Hahn and Schnattinger, 1998; Maedche and Staab, 2001; Craven et al., 2000; Faure and Nedellec, 1998).

In this paper, we focused on the second approach and introduced a hybrid method to learn ontologies from natural language texts. In this section, we will first have an

[18] As we had no prior lexical and ontological knowledge, in these case studies we restricted the grammar of input texts to cover simple sentences to start learning from. Repeating the experiments on learning from scratch using real texts showed about 30–50% decreasing in precision.

overview of related works in the field of ontology learning and will extract a framework with six main dimensions to describe the ontology learning systems' features. Then we will summarize the features of Hasti according to this framework and at last will compare it to some of the closest systems to it.

6.1. *Features of ontology learning systems*

In the field of ontology learning, some works have been done during the last two decades. Some of them such as Mikrokosmos (Nierenburg et al., 1996) and Cyc (Lenat and Guha, 1990) acquire ontological knowledge manually and then after building a huge general ontology allow to acquire more knowledge in a semiautomatic manner based on the huge base ontology. Some others choose a limited domain and extract knowledge from texts in the selected domain (Yamaguchi, 2001). Some projects focus on learning ontological knowledge having a predefined lexicon (Kietz et al., 2000), while some others learn lexical knowledge for new words too (Thompson and Mooney, 1999). Developed systems may extract just taxonomic knowledge (Hahn and Schnattinger, 1998; Suryanto and Compton, 2000; Heyer et al., 2001; Sundblad, 2002), discover non-taxonomic conceptual relations as well (Maedche and Staab, 2000a; Agirre et al., 2000), discover specific roles such as learning subject and objects of verbs by Pereira et al. (1993), or extract knowledge of a specific word type for example discovering verb frames by Faure and Nedellec (1998), inferring verb meanings by Wiemer-Hastings et al. (1998), classifying adjectives by Assadi (1997) and identifying names by Bikel et al. (1999).

On the other hand, projects on learning ontologies, work on different ontological structure and topology with different approaches. Strict hierarchy of concepts (just one parent for each node) (Emde and Wettschereck, 1996), pyramid hierarchy (no two links crossing) (Faure et al., 1998), directed graphs (Maedche and Staab, 2000b) and conjunction of directed graphs and axioms (such as Hasti) are some of the used structures.

Learning ontologies via statistical methods such as Wagner (2000) and learning by collocations and co-occurrences (Yamaguchi, 2001), logical approaches such as inductive logic programming (ILP) (Zelle and Mooney, 1993), FOL-based clustering methods (Bisson, 1992), and proposition learning (Bowers et al., 2000), Linguistic-based approaches such as syntactic analysis (Faure et al., 1998) and morpho-syntactic analysis (Assadi, 1997), and using pattern parsing (Finkelstein-Landau and Morin, 1999; Vargas-Vera et al., 2001) are some of the introduced approaches to learn ontological knowledge from texts. There are also some projects, which apply a combination of different approaches to learn different components of ontologies such as using association rules, formal concept analysis and clustering techniques in Maedche and Staab (2001) or combining FOL rule learning with Bayesian learning by Craven et al. (2000).

In general, the major distinguishing factors between ontology learning systems are (Shamsfard, 2002; Shamsfard and Barforoush, 2003) as follows.

(1) The learned elements (concepts, relations, axioms, instances and syntactic categories and thematic roles).

(2) Starting point (prior knowledge and the type and language of input).
(3) Preprocessing: (linguistic processing such as deep understanding or shallow text processing).
(4) Learning Method consisting of
    (a) Learning approach (statistical, logical, linguistic-based, pattern matching, template driven and hybrid methods).
    (b) Learning task (classification, clustering, rule learning, concept formation, ontology population).
    (c) Learning category (supervised vs. unsupervised, on-line vs. offline).
    (d) Degree of automation (manual, semi-automatic, cooperative, full automatic).
        (i) Type and amount of user intervention.
(5) The resulted ontology: (features such as coverage degree, usage or purpose, content type, structure and topology and representation language).
(6) Evaluation methods (evaluating the learning methods or evaluating the resulted ontology).

As the paper presented, features of Hasti according to the above framework are as the following.

1. *The learned elements*: Hasti learns words, concepts, relations (both taxonomic and non-taxonomic) and some axioms. It also discovers words' syntactic categories and case roles.
2. *Starting point*: Hasti starts from a small ontology kernel. There is no need to a large base ontology or a lexical knowledge base. Its lexicon and ontology are nearly empty at the beginning. In other words, Hasti builds ontologies from scratch. By scratch we mean there is no general or domain ontology and no semantic lexicon to help the learning process. The base knowledge base just contains the learning algorithms and some linguistic knowledge such as the grammar, morphological rules, syntactic and semantic templates and also the inference rules for logical reasoning. Any other knowledge will be learnt.
3. *Preprocessing*: The linguistic preprocessing on Persian input texts is a morpho-syntactic processing.
4. *Learning method*: The Learning approach in Hasti is a symbolic, hybrid approach, a combination of logical, linguistic-based, pattern matching and template-driven, semantic analysis and heuristic methods. It does concept learning and clustering in two modes: cooperative and full automatic (unsupervised) and also in on-line and offline modes. In the cooperative mode user intervene in naming concepts, determining weights, validating the artificial sentences built by the system and confirming the system's decisions.
5. *The resulted ontology*: It may be a general or domain-specific ontology according to the input texts. It is dynamic and flexible to changes. Its structure is a directed graph plus axioms and would be represented in KIF.
6. *Evaluation*: We used an empirical method to evaluate our approach by evaluating the resulted ontology. Hasti is tested on small-scaled texts and the built ontology has been compared with the ontologies built by people (experts and non-experts).

6.2. Comparison with related works

As we had a survey on related works, there is no work, which entirely matches the features and capabilities of Hasti. Instead there are some works, which are partially close to some parts of Hasti or do some common tasks or have some common features with Hasti. The general differences between Hasti and all related works are lacking of conceptual and lexical initial knowledge, ability to learn axioms, processing Persian texts and applying the specified hybrid learning approach. Each of the related works has also its own differences with Hasti, which some among others will be discussed in this subsection.

Hasti is close to the part of SynDiKATe (Hahn and Romacker, 2000), which is described in Hahn and Schnattinger (1998) since both do incremental clustering and both generate hypotheses from linguistic and conceptual evidences, but their methods to do these tasks are different. SynDiKATe uses linguistic and conceptual quality labels to score concept hypotheses about the place of concept in the hierarchy and does a quality based classification to find the proper hypothesis, while Hasti finds the appropriate place by moving downward from the original father to its *Msf* (cf. Section 4.3.1). SynDiKATe also extracts some non-taxonomic relations (such as case roles) from semantic interpretation of texts, which is similar to a part of relation making between ontels in the knowledge extractor of Hasti. But Hasti does offline clustering and ontology reorganization too. There are also differences in their concept learning, hypothesis reduction and clustering algorithms. Moreover, the focus of Hasti is learning ontological and lexical knowledge from scratch while the focus of SynDiKATe is to extract domain knowledge by text understanding upon general and domain ontologies and a semantic lexicon.

Hasti is somehow close to text-to-onto (Maedche and Staab, 2001) as both learn concepts, taxonomic and non-taxonomic relations using a multi-strategy method. Although the architectures are similar, but there are many differences between these projects such as the following.

The input in Hasti is Persian texts while in text-to-onto may be German texts or (semi) structured data such as DTDs or XML texts. text-to-onto does not cluster concepts incrementally; it rather does an offline clustering at the end of preprocessing the entire input. The multi-strategy learning method in text-to-onto is a combination of association rules, formal concept analysis and clustering and it uses different learning algorithms, for example in learning non-taxonomic relations text-to-onto analyses co-occurrence frequencies (Maedche et al., 2002) while we do semantic analysis. text-to-onto uses binary relations while the relations in Hasti are not restricted and may be *n*-ary (for $n \geqslant 2$).

A part of Hasti, which extracts case frames for verbs, is related to projects such as wolfie (Thompson and Mooney, 1999) and asium (Faure et al., 1998). Hasti and wolfie learn semantic roles and lexicon while asium learns syntactic roles (subcategorization frames). Learned knowledge is represented by lists of attribute-values defining concepts in wolfie and by P-role relations between concepts in Hasti. Moreover, input sentences in wolfie are annotated by semantic labels while in Hasti and ASIUM input is natural text and labels are provided after learning phase. The proposed approach in ASIUM is suitable for technical corpora while ours may be used for general and specific domains with no cost of template changes. ASIUM only uses head nouns while we consider all words in Hasti.

On the other hand, some of the templates used in Hasti, are close to lexico-syntactic patterns in the work by Hearst (1992), but they (in Hasti) are defined to extract wider range of relations. Hearst's patterns which are also used in Hasti, aim at extracting hyponymy relations but our syntactic and semantic templates are used to extract syntagmatic and paradigmatic relations such as thematic roles (verb frames), hyperonymy/hyponymy, synonymy, meronymy, etc. Our templates are general and domain/application independent while some related works (Finkelstein and Morin, 1999) acquire and use semantic patterns, which are fully domain dependant.

7. Conclusion and further works

Automatic learning of ontologies is a solution to ontology creation and management problems. This paper described a new approach to learn ontologies from scratch and introduced Hasti as a test bed for our model of automatic ontology building. Hasti is an ontology learning system, which builds ontologies upon a small kernel. It learns lexical and ontological knowledge from Persian texts. Its ontology contains concepts, taxonomic and non-taxonomic conceptual relations and axioms, all codified in a simplified version of KIF.

Hasti uses symbolic ontology learning methods. Its learning approach is a hybrid approach, a combination of logical, linguistic-based, template-driven and semantic analysis methods with several usage of heuristics.

The logical approach is applied by the inference engine of the knowledge extractor component. It performs some logical inferences on the knowledge base to deduce new knowledge (new relations between concepts and new axioms) and also to find the related terms through the properties of relations such as transition, symmetry, etc.

The linguistic-based approach is applied by the NLP component. It does morpho-syntactic analysis to process input texts and extract case roles. It also uses syntactic templates (in the SST builder) which are themselves linguistic-based. So the NLP component use both linguistic-based and template-driven methods.

The template-driven approach is applied not only by the SST builder, but also by the ontel creator. The ontel creator uses semantic templates to extract conceptual knowledge. These templates are based on the semantic of sentences. In other words, some semantic analysis is done to extract the required knowledge.

So the semantic analysis approach is applied by both the knowledge extractor and the ontology manager. The ontology manager places ontels, clusters, reorganizes and refines the ontology according to semantic similarity of ontels.

On the other hand, in various parts of the learning process heuristics are used to eliminate ambiguities, decrease the hypothesis space and choose a good candidate among others. For example, the heuristics for choosing the best group/pair to be merged in clustering concepts may be the first pair, the pair with maximum common parents or the pair with at least one middle-concept for the concept-based algorithm and the group with maximum members for the relation-based algorithm.

In the cooperative mode, user may intervene in naming concepts, determining weights, validating the artificial sentences built by the test generator and confirming the system's decisions. In the automatic mode concepts' names[19] and weights are determined by the system, generated tests will be kept to be searched for in further sentences and the system uses default reasoning and closed world assumption to decide whenever needed.

Although the proposed approach is tested for Persian texts, but as the kernel is language neutral, the method is adaptable for other languages by changing the linguistic knowledge such as the grammar, morphological rules and some few parts of syntactic and semantic templates. It is domain/application independent and learns lexical and ontological knowledge for general and specific domains.

Further works to complete the project are performing more tests on large corpora for various learning algorithms and evaluating the system by standard measures (such as precision and recall). Enhancing the hypothesis reduction procedure, working on automating weight assignment and improving axiom learning methods to extract and learn implicit axioms in text are other further works for Hasti. Automatic adjustment of system parameters such as offline clustering period (P) and efficient merge and split thresholds are other improvement which should be done for Hasti.

On the other hand, enhancement of the NLP component consisting of language expansion (expanding the grammar and the linguistic knowledge) to cover wider range of sentences, and adding ability to learn morphological constituents are in the plan of future works.

We also plan to work on compound-concept building. Sometimes it is more efficient to have an adjective or noun phrase as the concept instead of separating the head noun from its adjective. These are the times in which the phrase occurs frequently in the text or is a significant concept in the domain. Using statistical methods to find these phrases can help to build compound concepts.

References

Agirre, E., Ansa, O., Hovy, E., Martínez, D., 2000. Enriching very large ontologies using the WWW. Proceedings of the ECAI 2000 Workshop on Ontology Learning (OL'2000), Berlin.

Assadi, H., 1997. Knowledge aquisition from texts: using an automatic clustering method based on noun-modifier relationship. Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL'97), Madrid, Spain.

Barforoush, A.A., Shamsfard, M., 1999. Hasti: a model of ontology for NLU systems. Proceedings of the 7th Iranian Conference on Electrical Engineering, Tehran, Iran, pp. 91–98.

Bikel, D.A., Schwartz, R., Weischedel, R., 1999. An algorithm that learns what's in a name. Machine Learning 34, 211–231.

[19] Hasti make a concept's name by concatenating its children's names.

Bisson, G., 1992. Learning in Fol with a similarity measure. Tenth National Conference on Artificial Intelligence (AAAI'92), AAAI Press, San Jose, CA, pp. 82–87.

Bowers, A., Giraud-Carrier, C., Lioyd, J., 2000. Classification of individuals with complex structure. Proceedings of the 17th International Conference on Machine Learning (ICML'2000), Stanford, US, June 29–July 2, pp. 81–88.

Connolly, D., van Harmelen, F., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A., 2001. DAML + OIL (March 2001) Reference Description, W3C Note 18 December 2001, available at: http://www.w3.org/TR/daml + oil-reference.

Craven, M., et al., 2000. Learning to extract symbolic knowledge from the world wide web. Artificial Intelligence 118, 69–113.

EAGLES, 1996. Word clustering. EAGLES preliminary recommendations on semantic encoding Interim Report available at: http://www.ilc.pi.cnr.it/EAGLES96/rep2/node37.html.

Emde, W., Wettschereck, D., 1996. Relational instance based learning. In: Saitta, L. (Ed.), Proceedings of the 13th International Conference on Machine Learning (ICML'96), Morgan Kaufmann, San Francisco, CA, pp. 122–130.

Faure, D., Nedellec, C., 1998. A corpus-based conceptual clustering method for verb frames and ontology acquisition. In: LREC Workshop on Adapting Lexical and Corpus Resources to Sublanguages and Applications, Granada, Spain.

Faure, D., Nedellec, C., Rouveirol, C., 1998. Acquisition of semantic knowledge using machine learning methods: the system ASIUM. Technical Report number ICS-TR-88-16. Laboratoire de Recherche en Informatique, Inference and Learning Group, Universite Paris, Sud.

FernÁndez, M., Gomez-Perez, A., Juristo, N. 1997. METHONTOLOGY: from ontological art towards ontological engineering. Workshop on Ontological Engineering. Spring Symposium Series. AAAI97, Stanford, USA.

Finkelstein-Landau, M., Morin, E., 1999. Extracting semantic relationships between terms: supervised vs. unsupervised methods. In: Proceedings of International Workshop on Ontological Engineering on the Global Information Infrastructure, Dagstuhl-Castle, Germany, pp. 71–80.

Genesereth, M.R., Fikes, R., 1992. Knowledge Interchange Format, Reference Manual, Version 3. Stanford University, CSD Tech-Report Logic 92-1.

Guarino, N., Carrara, M., Giaretta, P., 1994. An ontology of meta-level categories. Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR'94). Morgan Kaufmann, San Mateo, CA.

Hahn, U., Romacker, M., 2000. The SynDiKATe text knowledge base generator. Data and Knowledge Analysis 35 (2), 137–159.

Hahn, U., Schnattinger, K., 1998. Towards text knowledge engineering. Proceedings of 15th National Conference on Artificial Intelligence (AAAI'98), Madison, WI, pp. 524–531.

Hearst, M.A., 1992. Automatic acquisition of hyponyms from large text corpora. Proceedings of the Fourteenth International Conference on Computational Linguistics, Nantes, France.

Heyer, G., Läuter, M., Quasthoff, U., Wittig, T., Wolff, C., 2001. Learning relations using collocations. Proceedings of the IJCAI 2001 Workshop on Ontology Learning (OL'2001), Seattle, USA.

Kent, R., 2002. The IFF Core Ontology. Available at: http://suo.ieee.org/IFF/versions/20020102/IFFCoreOntology.pdf.

Kietz, J., Maedche, A., Volz, R., 2000. A method for semi-automatic ontology acquisition from a corporate intranet. Proceedings of the EKAW'2000 Workshop on Ontologies and Texts, France.

Lacher, M.S., Groh, G., 2001. Facilitating the exchange of explicit knowledge through ontology mappings. Proceedings of FLAIRS'2001, AAAI Press, Menlo Park, CA.

Lenat, D.B., Guha, R.V., 1990. Building Large Knowledge Based Systems. Representation and Inference in the CYC Project. Addison-Wesley, Reading, MA.

Maedche, A., Staab, S., 2000a. Semi-automatic engineering of ontologies from text. Proceedings of the twelth International Conference on Software Engineering and Knowledge Engineering (SEKE'2000), Chicago.

Maedche, A., Staab, S., 2000b. Discovering conceptual relations from text. In: Horn, W. (Ed.), Proceedings of the 14th European Conference on Artificial Intelligence (ECAI'2000). IOS Press, Berlin, Germany, pp. 321–325.

Maedche, A., Staab, S., 2001. Ontology learning for the semantic web. IEEE Intelligent Systems 16 (2), 72–79.

Maedche, A., Pekar, V., Staab, S., 2002. Ontology Learning Part One: Learning Taxonomic Relations. Available at: http://wim.fzi.de/wim/publications/entries/1016618323.pdf.

Miller, G.A., 1995. Wordnet: a lexical database for English. Communications of the ACM 38 (11), 39–41.

Nierenburg, S., Levin, L., 1992. Syntax driven and ontology driven lexical semantics. In: Pustejovsky, J., Bergler, S. (Eds.), Lexical Semantics and Knowledge Representation. LNAI, Vol. 627. Springer, Berlin, pp. 5–20.

Nierenburg, S., Beale, S., Mahesh, K., Onyshkevych, B., Raskin, V., Viegas, E., Wilks, Y., Zajac, R., 1996. Lexicons in the mikrokosmos project. Proceedings of AISB Workshop on Multilinguality in the Lexicon, Brighton, UK, pp. 26–33.

Noy, N.F., Musen, M.A., 2000. PROMPT: algorithm and tool for automated ontology merging and alignment. Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000), Austin, TX.

Pereira, F., Tishby, N., Lee, L., 1993. Distributional clustering of English words. In: Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL'93), pp. 183–190.

Ryutaro, I., Hideaki, T., Shinichi, H., 2001. Rule induction for concept hierarchy alignment. Proceedings of the IJCAI 2001 Workshop on Ontology Learning (OL'2001), Seattle, USA.

Shamsfard, M., 2002. A framework for classifying and comparing ontology learning systems. Technical Report No. 81-4001-107. Intelligent Systems Lab., Computer Engineering Dept., AmirKabir University of Technology. Available at http://www.pnu.ac.ir/~shams/tech-81-107.pdf.

Shamsfard, M., 2003. Design of ontology learning model, prototyping in a Farsi text understanding system. Ph.D. Dissertation, Computer Engineering Dept., AmirKabir University of Technology, Tehran, Iran, January 2003.

Shamsfard, M., Barforoush, A.A., 2000. A basis for evolutionary ontology construction. Proceedings of 18th IASTED International Conference on Applied Informatics (AI'2000), Innsbruck, Austria.

Shamsfard, M., Barforoush, A.A., 2001. Computational lexicon: the central structure in NLP systems. AmirKabir Journal of Science and Technology 12 (48), 449–462 AmirKabir University Press, Tehran, Iran.

Shamsfard, M., Barforoush, A.A., 2003. The state of the art in ontology learning: a framework for comparison. Knowledge Engineering Review, To be appeared.

Sowa, J.F., 1995. Top level ontological categories. International Journal of Human Computer Studies 43, 669–685.

Sundblad, H., 2002. Automatic acquisition of hyponyms and meronyms from question corpora. Proceedings of the ECAI 2002 Workshop on Machine Learning and Natural Language Processing for Ontology Engineering (OLT'2002), Lyon, France.

Suryanto, H., Compton, P., 2000. Learning classification taxonomies from a classification knowledge based system. Proceedings of the ECAI 2000 Workshop on Ontology Learning (OL'2000).

Thompson, C.A., Mooney, R.J., 1999. Automatic construction of semantic lexicons for learning natural language interfaces. Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'99), Orlando, FL, pp. 487–493.

Vargas-Vera, M., Domingue, J., Kalfoglou, Y., Motta, E., Buckingham, S., 2001. Template-driven information extraction for populating ontologies. Proceedings of IJCAI'2001 Workshop on Ontology Learning (OL'2001), Seattle, USA.

Wagner, A., 2000. Enriching a lexical semantic net with selectional preferences by means of statistical corpus analysis. Proceedings of ECAI'2000 Workshop on Ontology Learning (OL'2000), Berlin, Germany.

Wiemer-Hastings, P., Graesser, A., Wiemer-Hastings, K., 1998. Inferring the meaning of verbs from context. Proceedings of the 20th Annual Conference of the Cognitive Science Society. Lawrence Erlbaum Associates, Mahwah, NJ.

Yamaguchi, T., 2001. Acquiring conceptual relations from domain-specific texts. Proceedings of IJCAI'2001 Workshop on Ontology Learning (OL'2001), Seattle, USA.

Zelle, Z.M., Mooney, R.J., 1993. Learning semantic grammars with constructive inductive logic programming. Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI'93), Washington, DC, pp. 817–822.