A Framework for Document-driven Evolution of Ontology

Krzysztof Węcel
The Poznań University of Economics
Department of Management Information Systems
Al. Niepodległości 10, 60-967 Poznań, Poland
K.Wecel@kie.ae.poznan.pl

Abstract

We present a framework for updating ontology as new documents arrive. The framework consists of three main layers: extraction, internal storage and refinement. In the extraction layer axioms are extracted from the text by analyzing named entities, contexts and relations. We introduce the notion of temporal axioms, i.e. the axioms that are valid in a certain time interval. This information is then used to reason about ontology evolution when those axioms are added to an ontology. We also present cases for evolution and corresponding changing axioms. In refinement layer, final ontology for a given point in time is built.

1. Introduction and Motivation

Business information should be complete, originate from a credible source and be up-to-date. Everyday business news reflects changes in the world. In information retrieval, users were interested in finding relevant documents. Current trends in research point at ontologies as a way to represent information in a structured way, thus facilitating more precise querying and question answering.

In a classical approach, ontologies represent a static world. Any change in the world requires a manual update in an ontology what is not adequate to the fast changing environment. The ontology that is not up to date has little value since it does not describe the world properly. Therefore, ontology evolution and maintenance is considered as more and more important issue. One of the approaches could be maintenance by means of extracting changes from selected relevant documents. In this paper, we propose a framework for document-driven evolution of ontology.

Only recently some of the frameworks have started to pay attention to time. In our system, time is considered as one of the dimensions that organizes information. It orders the changes in the ontology, allows checking the validity of the facts in time and verifying consistency of ontology in time.

As a result the framework delivers: up-to-date ontology (static aspect), which is useful for retrieving knowledge, and a report on changes to ontology (dynamic aspect), which is useful for observing trends.

2. Related Work

In order to execute structured queries on text we need to employ information extraction techniques. Although the problem of extracting information is well known and many approaches have already been proposed, there is no satisfactory application for ontology evolution. There are still many challenges that hinder precise extraction of meaning. On the one hand, knowledge in documents is imprecise; on the other hand, ontology learning algorithms generate uncertain knowledge. The uncertainty of learning algorithms may be ascribed to two types of reasons: language-related and content-related. The first comprises ambiguity of language, metaphors, complex syntax, which contemporary language tools cannot handle; the latter - addressed by our framework - incomplete information, imprecise information, false information, subjectiveness (opinions, beliefs), which cannot be verified without checking in many sources.

Annotation is simply additional information (metadata) associated with a fragment of a document and is one of the ways to structure the document. First attempts focused on named entity recognition (NER), which detects and classifies entities according to a predefined entity type system. Then a focus moved from concepts and their instances towards relations.

In order to compare the results of different approaches to structuring text, various evaluation schemes were developed. One of them, the Automatic Content Extraction (ACE), was created by the National Institute of Standards and Technology (NIST) [1]. ACE[1] consists of four tasks: (1) recognizing entities, (2) recognizing relations, (3) recognizing events, and (4) recognizing time expressions. The evaluation scheme initially consisted of Entity Detection and Tracking (EDT). Then, Relation Detection and Characterization (RDC) task was added and in 2004 Time Expression Recognition and Normalization (TERN) was introduced.

Several domain ontologies have been developed for document annotation purposes. The most important in the proposed framework is Ontology of Time . Some of the authors propose to merge document annotation and ontology learning [3].

Many different inspirations for several projects are taken from ACE. One of such projects is ONTOTEXT [4]. Most important features of this complex system are repository of facts, traceability, temporal binding of information, confidence of extracted information, verification of consistency.

---

[1] http://www.nist.gov/speech/tests/ace/index.htm

Another very similar system is Text2Onto [5], which aims at ontology learning. Ontology learning tasks were divided into concept extraction, instance extraction, similarity extraction, concept classification, and instance classification. The specific features of this system are incremental ontology learning, storing ontology in a proprietary format, and storing evidence's references to the source text.

3. Ontology Evolution

Before it is possible to analyze ontology evolution, one has to learn the initial ontology. There are many approaches to ontology learning [6] and most of them employ information extraction techniques; some rely purely on IR techniques over bag-of-words representation. When appropriate information is extracted, it should be compared to the existing knowledge. Since the knowledge may change in time, the occurrence of change should be located on time axis. Further on we analyze how to identify a change, what modeling primitives may be affected by the change and how to present the outcomes to the user.

3.1. Ontology Learning

In order to extract particular kind of information, one typically has to write extraction rules (patterns). The rules recognize a fixed set of entities classified according to the known schema. Principally, one needs a type system with some constraints, which may be called ontology. In the real-world applications one require much richer collection of classes and attributes, and may wish to detect if new types of entities emerge. Therefore, Ontology Learning (OL) emerged.

Ontology population focuses on instances while ontology learning on concepts and relations. Proper recognition of instances requires precise identification of concepts, thus ontology requires maintenance. The goal of annotating the document is actually to extract useful information that may be then useful for query answering. Instead of analyzing documents every time the query is submitted, one may store all information in a knowledge base. Such knowledge base may require updates very often and this is another argument for ontology maintenance.

In order to handle ontology evolution properly, one has to recognize time expressions very precisely ([7], [8]). Not only change has to be detected but also the time when it occurred has to be learned from the analyzed text. This information is not always explicit and tends to be ambiguous even for humans.

On the one hand, the goal is to extract information from document for ontology. On the other hand, we need the underlying ontology in order to know what should be extracted. Those two processes are interrelated and they support each other.

The following sources of ontologies required for ontology evolution have been identified:

- seed ontology (domain ontology), defines core entities that should be extracted and learned from text
- language ontology, facilitates the extraction of information from text, appropriate concepts are selected for domain ontology; may be used directly to grow the seed ontology
- documents, bring new information that has to be structured.

Usually, annotation of web resources relies on ontology, which defines concepts and relations. Due to the still evolving world, it is not feasible to update this ontology manually. Therefore, it is necessary to update ontology when text contains entities that cannot be annotated using known ontology. Thus, the process of annotation and ontology learning should be merged. Ontology should be rebuilt according to changes found in annotated documents. Moreover, documents should not only be annotated but extracted information should be stored in the database to allow further reasoning.

3.2. Principles for Ontology Change

Let us starting by analyzing what may change from very general point of view. When new documents are filtered to the system, they bring some facts. Those facts may have the following influence on knowledge base (cases):

- confirmation, new information confirms or supports known information, fact remains unchanged
- completion, new information add new details about existing entity, e.g. we knew that company X is located in city Y, and the new fact is that it is located on the street Z
- correction, new information changes known facts, e.g. when company X is indeed operating in city Y, not in city Z
- contradiction, new information contradicts old information, e.g. we knew that company X had invested in Y, and the new fact is that it did not invest in Y
- prolongation, new information extends the time validity of the known fact, e.g. “the president was elected for the next term”
- invalidation, the fact that was known to be true in the past is not true from a certain date, e.g. “the factory has just stopped producing cars”
- update, new information makes known facts up-to-date, e.g. we know the turnover for May and the new fact is turnover for June.

In order to represent those changes in the ontology we have to consider how static facts are represented. By looking at basic modeling primitives, the following changes have been identified:

- class: a new class added, an existing class removed, change in hierarchy of classes, a disjoint or equivalent class defined
- instances: instance re-classified, properties of instance added or removed
- relations: domain changed, ranges changed, relation instances changed, value of the property changed.

In order to handle ontology evolution we need a granule of change. In OWL-DL (which is the target formalism) such granule is an axiom, e.g. "X sub-class-of Y". Therefore, new facts are transformed into appropriate axioms. The list above does not contain all OWL axioms, e.g. inverse object properties, sub properties, since in the first attempt we focus on the restricted subset.

We claim that representation in form of axioms is not sufficient for business information: if it is unknown when a change occurred, it is not possible to reason about evolution. The business domain does not model a static world. Some of the cases, like prolongation, invalidation or update, use the notion of time. The axiom itself is useless unless it is bound to time, i.e. the period of validity is defined; hence the extension of formalism and motivation for the framework proposed in this paper. We define *temporal axioms* as axioms that are true only in certain time interval.

The formalism is required in order to:

- represent extracted information in a structural way
- compare new facts with knowledge base
- apply changes to the ontology if necessary.

3.3. Modeling Primitives

The most common modeling primitives are classes, instances, and relations.

3.3.1. Instances

Instances are the primary modeling primitive - most of the efforts focus on their recognition, they are classified, take part in relations, and when a query is formulated, one usually asks about individuals, and rarely about a structure of the reality. As a prerequisite, we need efficient methods for extraction and annotation of named entities. As open issue remains how precise the information extraction methods should be. Some research show that even between different human annotators there is often no consensus on how some resources should be annotated.

Instances extracted from text may play two roles: be just a named entity or provide a context for recognition of other named entities (e.g. location, time, organization). According to MUC, named entity recognition is considered a solved problem and the highest-scoring systems achieve precision of 97%, what is comparable to human performance [9].

Instances actually do not evolve - either they exist in a given time or not. What evolves, are only their relations to other instances in a certain period. One only has to recognize when such an entity was created. The way an entity is brought into being depends on the type of entity itself and is language-dependent, e.g. a human was *born*, a good was *produced*, an organization was *founded*. This task requires an underlying ontology that defines events that start and finish existence of an instance. Instances may take part in relation only when they really exist, therefore asserting the period of existence is crucial.

Finding only changes in instances is actually ontology population, we therefore focus also on classes.

3.3.2. Classes

An important issue with classes learned from the ontology is the required high precision of learning. Intuitively, when an instance is learned and recognized improperly, it is not that harmful compared to a class that is learned improperly. Therefore, particular attention should be paid to correctness of terminological knowledge (like TBox in Description Logic); assertional knowledge (ABox) affects smaller number of entities.

The core issue of ontology evolution is how to treat classes and handle changes in classes. A class should exist as long as it has any instance. Description logic distinguishes two kinds of classes: primitive (only necessary conditions are specified) and defined classes (both necessary and sufficient conditions), and there are two hierarchies of classes: asserted and inferred. It may also happen that class has to be removed from the ontology and therefore instances should be reclassified, e.g. when a certain kind of organizations is removed from the law.

We have to consider whether the goal of text analysis is actually the creation of ontology from scratches or rather refinement of existing ontology. To some extent, we can use such sources of information as WordNet. When analyzing text in natural language this kind of language ontology is the most useful, at least for disambiguation purposes. Significant part of the language ontology may be reused, and domain-specific parts may be learned from analyzed documents.

The question remains if an algorithm exists that is capable of creating or identifying *defined classes* automatically. Such a task done automatically is prone to errors and for that reason an underlying domain ontology is indispensable. Moreover, in order to achieve high precision of class recognition we suggest employing user feedback.

3.3.3. Relations

As extraction of instances is relatively well researched, most of the effort is now put on relations. For example, ACE defines extraction of relations as one of its four tasks. Unfortunately, the experience of ACE shows that finding relations is a challenging task.

In the proposed framework there are two issues that go beyond the problems defined for text annotation. Firstly, the set of relations to be extracted is not fixed like in ACE. Here, a relation should be referred to as object property when using OWL-DL. Some simple relations, like defined by ACE program, can be extracted with some hard-coded rules. For ontology evolution to take real advantages, we need much more flexible approach. Because the final ontology is to be exported in OWL-DL and we plan to use description logic reasoners, also some properties of the relation should be known: whether the relation is

symmetric, transitive, functional or inverse functional and what is its inverse relation. Some of the properties probably cannot be determined automatically with satisfying confidence, e.g. cardinality, domain, and range restrictions. Again, the domain ontology is essential.

Secondly, a need to determine in which period this relation is valid makes the task even harder. This kind of information is not to be found in domain ontology and has to be extracted from text. Moreover, in order to take part in relations both entities has to exist throughout the interval of relation's validity. Again, time context proves to be an important issue. The next section studies in details temporal relations.

3.4. Time-related Issues

In the framework, we are interested in the evolution of ontology in time (time as an attribute of an axiom), and not in the modeling of temporal aspects in the ontology itself.

Time context is inherent to the evolution. Many of the axioms are only valid during the limited period. Therefore, it is not sufficient to extract facts from the documents but also those facts should be bound to time. In some cases the period when the information should be valid is commonly known (e.g. 5 years for a president of a country), in other cases not (e.g. open period for existence of music band).

Knowledge may change expectedly (e.g. at end of fiscal year) or unexpectedly (most other cases). Additionally, some expected events may occur earlier and thus become unexpected events. In the first case, people anticipate the change, it is possible to handle it; in the latter case, it is usually not simple to figure out that the change occurred and when.

General relations, like "Employee works for Company", extracted from text may be useful for defining domain and range of relations, thus providing evidences for ontology evolution.

In order to describe the relation instance one has to find not only instances but also determine when a given relation instance is in force. Binding to time is helpful in checking some constraints on relations, like cardinality. For example, two sentences: "Warsaw is the capital of Poland" and "Cracow is the capital of Poland" do not contradict provided that the second sentence is written in 1500. In a static approach, it is asserted in the ontology that one country has only one capital. In the evolving world, we need to assert that a country may have only one capital at a time but several capitals throughout the whole history. Generally, we have to distinguish between uniqueness (one value at a time) and universal uniqueness (one value at all). Attributes that are universally unique never change (e.g. place of birth); hence, they need not be bound to time.

The universally unique relations are particularly useful for finding contradictions in documents. In unique relations, when there are contradictory facts but stated on different dates one have to change the time validity of the first one and add another axiom with *updated* fact (e.g. when somebody changes a

job). Sometimes it is not possible to give the exact date of change and therefore fuzzy representation should be used, leading to different confidence levels.

To sum up, the ontology should contain information on:

- invariability of some relations
- cardinality in time context.

3.5. Detection and Presentation of the Evolution

In order to discover the need for a change one has to analyze new facts extracted from incoming documents. After extraction and formalization in form of axioms, it is possible to compare them to current state of knowledge base. The cases are presented in the Table 1. The notation is as follows: P is a predicate, P_{T1} is a temporal predicate that is valid in interval T1, meets, overlaps, before are taken from interval calculus [10].

Table 1. Cases for evolution and corresponding changing axioms.

Type of change	**Existing axioms**	**Removed axioms**	**Added axioms**
confirmation	P(x,y)	-	-
completion	$P_1(x,y)$	-	$P_1(x,z) \wedge \text{subclassof}(z,y) \vee P_2(x,z)$
correction	P(x,y)	P(x,y)	P(x,z)
contradiction	P(x,y)	P(x,y)	-
prolongation	$P_{T1}(x,y)$	-	$P_{T2}(x,y) \wedge \text{meets}(T1,T2)$
invalidation	$P_{T1}(x,y)$	$P_{T1}(x,y)$	$P_{T2}(x,y) \wedge \text{overlaps}(T1,T2)$
update	$P_{T1}(x,y)$	-	$P_{T2}(x,z) \wedge \text{before}(T1,T2)$

First, a new fact has to be classified to one of the cases mentioned above to allow proper intervention. Classification of the fact to one of the cases also depends on how it can be resolved. In confirmation case, the confidence of the given axiom should be increased. In any other case, new axioms are introduced into ontology, which may become inconsistent. In some cases the new fact directly contradicts with a known fact, otherwise one needs a logical formalism in order to infer the contradiction.

There are different possible approaches to response to a change. Some of the approaches just put the fact into the knowledge base and resolve potential contradictions when final ontology is requested, e.g. Text2Onto. Such an approach is not very useful for applications that require prompt action. Therefore, in the proposed framework the reaction for new fact is immediate; triggers may be set up to allow further actions.

There are different approaches to handle ontology change that may lead to inconsistency. The following major use cases has been identified in [11]:

- maintaining consistency of initially consistent ontology by applying appropriate changes

- repairing an inconsistent ontology
- reasoning with inconsistent ontology (usually on consistent sub-ontology)
- finding the right version of an ontology that is consistent.

We claim that it is preferred to keep ontology consistent (with regards to some moment in time) in order to react properly to consecutive changes extracted from the stream of text documents. Inconsistency may be obvious, e.g. one fact says A and the other ¬A, but more often reasoning is required. We do not believe that very complex reasoning may be executed for the whole domain (with temporal logics in background, which may be undecidable). We focus only on contradicting facts with shallow reasoning. Deeper inconsistencies may be discovered just before the final ontology is created in OWL-DL, i.e. when time is not considered. There are also some approaches that allow to debug and repair an inconsistent ontology [12].

4. Framework

Having analyzed the requirements, we propose the following framework for document-driven ontology evolution. The model is organized into a number of layers. Each layer is responsible for one of the steps in handling the evolution. Each layer holds specific type of data and makes this data accessible to the higher layers. An overview of the developed framework is presented in Figure 1.

We have identified three main layers required for evolution: the bottom one is responsible for extraction of information from text. Structured representation of the documents is stored in an internal format in the middle layer. The top layer is responsible for producing final ontology that conforms to some general and user requirements.

4.1. Layers in the Stack

The bottom layer represents text that has to be analyzed. Text originates from documents that are filtered from pre-defined Internet sources. By applying information extraction techniques, we are able to structure those documents into contexts (temporal and spatial) and extract named entities (NEs). Then relations among named entities are discovered. Additional information from ontology learned so far is utilized as well as information about contexts what helps further in disambiguating NEs and resolving co-references.

For ontology evolution it is crucial to assert when given relations are supposed to be valid. Therefore, using the information from time context, temporal relations are composed, i.e. relations with assigned intervals or instants of their existence. It is not always possible to define appropriate interval if such information is not given explicitly in text. Only temporal relations may be an evidence for the change in ontology.

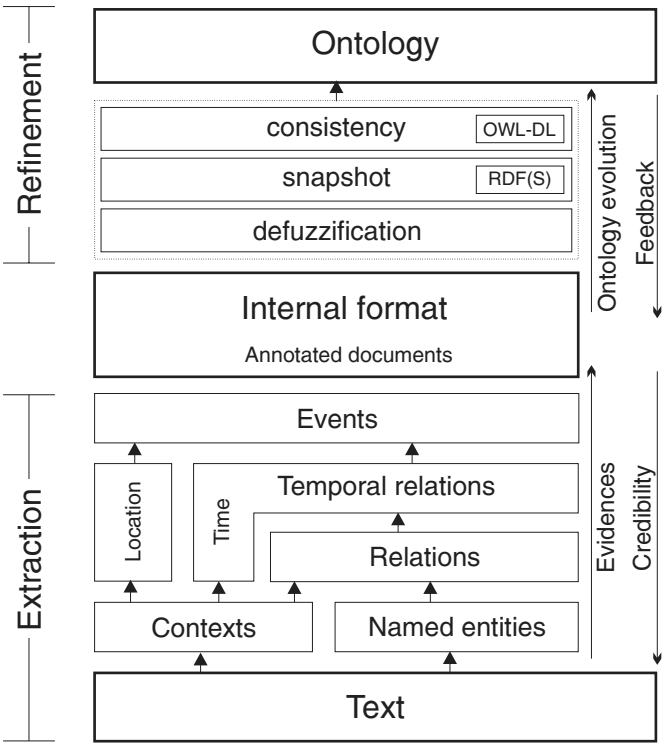

Figure 1. A Framework for Ontology Evolution from Documents.

The next layer extracts events that occur in a given location and in a given time (contexts are used). Unlike the relations, events directly create a need to change the ontology. For example, compare two statements: "somebody owns a company" and "somebody bought a company". In case of relations, one have to compare the knowledge base with new information found in the document; the evidence will either support the existing fact or falsify it. Relations and events may be distinguished by looking at a verb: some verbs represent a state and some an action, and it may be guessed by looking in language ontology for roles of verbs. Separate patterns should be used for extraction of events.

From each text several evidences may be extracted. They are stored in the middle layer in the internal format. Throughout the whole process documents are annotated in each layer and therefore any kind of tool may be used to recognize named entities, contexts, or relations (modular approach).

The middle layer uses modeling primitives to represent information extracted from text. These are in the form of axioms about concepts, instances and relations with certain degree of confidence and with temporal information about their validity. We have decided to use a generic relational database model in order to store axioms (therefore no temporal modeling in ontologies itself). In such a way, it is possible to represent uncertainty and then produce ontology on

demand according to some general requirements (e.g. consistency) and user needs (e.g. query). References of evidences back to text are also preserved in the database. Therefore, when a document is evaluated as unreliable all of its evidences should have weights lowered (even to zero). Some evidences may point to several documents. Evidences collected over a longer period allow to achieve higher confidence.

Ontology is produced in the remaining layers. The three layers - defuzzification, snapshot, and consistency - are ordered, however, they may support each other in several iterations to produce the final ontology. Because in the internal format each axiom has a confidence level, it is not very useful for reasoning. In defuzzification, certain threshold is accepted and only the most certain axioms are an input to the next layer. Axioms have also temporal information on their validity and user might be interested only in the state of the ontology on particular date. Therefore, different versions of ontology may be created. Finally, consistency should be evaluated. Only after removing uncertainty and temporal information we are able to reason about the consistency of the ontology, when the set of axioms is fixed and any existing Description Logics reasoner may be used. If the resulting ontology is not consistent or not complete, the process of repairing ontology should be started taking into account information from two preceding layers. The algorithms are to be developed.

The last layer is responsible for exporting the ontology in the selected language with appropriate knowledge representation formalism.

4.2. Processes

There are four main processes in the framework (see Figure 1 on the right). Two of them are responsible for transforming crude text at the bottom into refined ontology at the top. The other two allow to improve the overall process of ontology learning and evolution by propagating the control down the stack.

4.2.1. Collecting Evidences

The process of collecting evidences from text was described in the previous section. What was not mentioned, are certain requirements for the algorithms – they should be able to handle incremental learning of ontology, i.e. adding and possibly removing evidences when new documents arrive or are invalidated. Changes in lower layers are propagated to upper layers.

4.2.2. Ontology Evolution

Having collected evidences from text, their influence on ontology should be evaluated. Inference process aims at checking the consistency of the ontology as well as for triggering appropriate changes in the ontology. Pre-defined consistency conditions and user-defined consistency conditions are taken into account.

4.2.3. Feedback

The consistency of ontology is checked by the system and user intervention is unnecessary. However, consistent ontology might not appropriately represent the world. Therefore, it is necessary to introduce a feedback mechanism, in which user may evaluate the truth of axioms or correct the ontology. Information about acceptance or rejection of axioms should be propagated down to other layers.

Main impact of the feedback in yes/no answer to the proposed change is first on defuzzification layer. When a user evaluates certain axiom, its confidence level is then either one or zero (we suppose that user is sure about her statements). The change in confidence has influence on higher as well as lower layers. The impact on higher levels is reflected in new ontology that has to be built. Moreover, users may also add other axioms and correct existing ones. Finding satisfactory solution may take several iterations.

Any change suggested by user should also be tracked down to the bottom layers of the stack; the impact on lower levels is described in the next section.

It has to be distinguished whether some of the facts from the document were false from the beginning of analyzed interval or were just updated later. In the first case, the confidence of axioms should be lowered. In the latter case, the temporal validity of axiom should be updated and the new axiom added. It also has to be checked whether an axiom was correct but out of scope of the snapshot. In such case the interval in temporal relation should be extended. More generally, another form of feedback user might give is about temporal validity of certain axioms. Such feedback information usually would not be useful for improving the algorithms as it is harder to track down in evidences. The question remains if the last task is computationally feasible.

4.2.4. Credibility

The actual goal of the users' feedback is not only to correct the ontology but also to improve the way the ontology is learned. We may achieve this by:

- identifying reliable documents and sources of documents
- identifying successful extraction patterns.

Based on user's feedback some of the axioms become fixed in the ontology. Therefore, documents or patterns that create those axioms are particularly valuable for learning. When a user gives a positive feedback such document or pattern should become more important, and respectively less important for negative feedback. Whether the credibility concerns a document or a pattern, should be decided by the user, who should look at the source text via links from evidences. It could also be possible to guess it based on evaluation of several documents and patterns given by a greater number of users.

Firstly, this process is particularly important for verification of the initial patterns given by a user, who may expect that some of them work, but in practice they cover too many or too few types of axioms. The correct extraction patterns

are strengthened and the incorrect ones are removed, ultimately leading to improvement of the ontology extraction algorithm.

Secondly, this process eliminates unreliable or hard-to-analyze sources of information. The default value for reliability of a document is the reliability of the source. When a greater number of documents from a given source have little importance, also significance of the source is decreased.

An appropriate user interface is required in order to facilitate the whole process of credibility evaluation. Interface has to map changes to evidences in text so the user may read fragments of source documents and make an evaluation.

Verification of sources and documents could also be done automatically based purely on confidence levels of extracted axioms. The algorithms are an open issue; an algorithm similar to PageRank for distribution of the credibility weights could be developed.

5. Conclusions and Future Work

This paper presented work in progress on the system that allows to extract information from a stream document for ontology evolution purposes. What makes this framework different is the role of time: every axiom is bound to time interval and changes are resolved in accordance to their time validity.

In the model, most fuzzy layers are located at the bottom (just after the extraction from text) and then the knowledge is refined in the upper layers. Such an approach enables a user to see the version of ontology from a given time and additionally to analyze cause-effect chains. One may question this approach as there are serious attempts to add fuzziness to OWL but robust reasoners does not exist yet.

The main feature of our approach is not to analyze thoroughly and represent each sentence but to extract only useful information (conforming to the given pattern). This approach allows avoiding many traps of current tools; imprecise information is just not extracted. We recognize only what is necessary and only what is unquestionable.

There are many goals that may be achieved with the framework: being up-to-date in the domain, acquiring knowledge, we also pointed at the possibility of ranking sources and documents by applying additional techniques.

Working application does not exist yet and many questions remain open as pointed in the paper. The work done so far focused on extraction, time issues and representation.

6. Acknowledgement

This research project has been supported by a Marie Curie Transfer of Knowledge Fellowship of the European Community's Sixth Framework Programme under contract number MTKD-CT-2004-509766 (enIRaF).

7. References

1. Doddington, G., et al., Automatic Content Extraction (ACE) program - task definitions and performance measures, in Proceedings of LREC 2004: Fourth International Conference on Language Resources and Evaluation. 2004.
2. Hobbs, J.R. and F. Pan, An Ontology of Time for the Semantic Web. ACM Transactions on Asian Language Information Processing, 2004. 3(1): p. 66-85.
3. Amardeilh, F., P. Laublet, and J.-L. Minel, Document annotation and ontology population from linguistic extractions, in Proceedings of the 3rd international conference on Knowledge capture. 2005, ACM Press: Banff, Alberta, Canada. p. 161-168.
4. Magnini, B., et al., From Text to Knowledge for the Semantic Web: the ONTOTEXT Project, in Semantic Web Applications and Perspectives, P. Bouquet and G. Tummarello, Editors. 2005, CEUR Workshop Proceedings: Trento.
5. Cimiano, P. and J. Völker, Text2Onto. A Framework for Ontology Learning and Data-driven Change Discovery, in Proceeding of NLDB'05. 2005.
6. Maedche, A., Ontology learning for the semantic Web. The Kluwer international series in engineering and computer science ; SECS 665. 2002, Boston: Kluwer Academic Publishers. xxiii, 244 p.
7. Ferro, L., et al., TIDES. 2005 Standard for the Annotation of Temporal Expressions. 2005, MITRE Corporation.
8. Mani, I., J. Pustejovsky, and R. Gaizauskas, eds. The Language of Time. 2005, Oxford University Press.
9. Grishman, R. and B. Sundheim, Message Understanding Conference-6: a brief history in Proceedings of the 16th conference on Computational linguistics - Volume 1 1996 Association for Computational Linguistics: Copenhagen, Denmark p. 466-471
10. Allen, J.F. and G. Ferguson, Actions and Events in Interval Temporal Logic. Journal of Logic and Computation, 1994.
11. Haase, P., et al., A Framework for Handling Inconsistency in Changing Ontologies, in 4th International Semantic Web Conference, ISWC 2005, Y. Gil, et al., Editors. 2005, Springer, LNCS: Galway.
12. Parsia, B., E. Sirin, and A. Kalyanpur, Debugging OWL ontologies, in Proceedings of the 14th international conference on World Wide Web. 2005, ACM Press: Chiba, Japan. p. 633-640.