Terminological Ontology Learning and Population using Latent Dirichlet Allocation

Francesco Colace and Massimo De Santo
Department of Information Technology
and Electronics Engineering
University of Salerno, ITALY
Email: {fcolace,desanto}@unisa.it

Luca Greco
Department of
Industrial Engineering
University of Salerno, ITALY
Email: lgreco@unisa.it

Vincenzo Moscato and Antonio Picariello
Department of
Informatica e Sistemistica
University of Naples ”Federico II”, ITALY
Email: {vmoscato,picus}@unisa.it

***Abstract*—The success of Semantic Web will heavily rely on the availability of formal ontologies to structure machine understanding data. However, there is still a lack of general methodologies for ontology automatic learning and population, i.e. the generation of domain ontologies from various kinds of resources by applying natural language processing and machine learning techniques In this paper, the authors present an ontology learning and population system that combines both statistical and semantic methodologies. Several experiments have been carried out, demonstrating the effectiveness of the proposed system.**

***Keywords*-Ontologies, Ontology Learning, Ontology Population, Latent Dirichlet Allocation**

I. Introduction

In the last decade many researchers have been involved in the development of methodologies for ontology definition, building, learning and population, due to the fact that ontologies are considered as an effective answer to the need of semantic interoperability among modern information systems: it is well known, in fact, that ontologies are the backbone of the Semantic Web and important means for sharing, reusing and reasoning about domain knowledge. Several theories have been developed, in different application domains and especially in the semantic web framework: however how to learn and populate ontologies is generally a not trivial and time consuming task and still remains an open research challenge.

The term “ontology learning” was introduced in [21] and can be described as the acquisition of a domain model from data. This process is historically connected to the introduction of the semantic web and needs input data from which to learn the concepts relevant for a given domain, their definitions as well as the relations holding between them. Ontologies can be learnt from various sources, be it databases, structured and unstructured documents or even existing preliminaries like dictionaries, taxonomies and directories.

With the explosion of information due to the Read/Write Web, ontology learning from text is becoming the most investigated in literature: ontology learning from text is the process of identifying terms, concepts, relations, and axioms from textual information and of using them in order to construct and maintain ontology [28]. In other words ontology learning from text is the process of deriving high level *concepts* and *relations* as well as *axioms* from information to form ontology.

Ontology Learning from text is generally composed by five phases that aim at returning five main outputs: *terms*, *concepts*, *taxonomic relations*, *non-taxonomic relations* and *axioms* [5].

To obtain each output, some tasks have to be accomplished and the techniques employed for each task may change among systems. In this sense the ontology learning process is really modular: in [28] the corresponding tasks and the plethora of employed techniques for each output are described.

The extraction of terms from text usually needs a *preprocessing phase* that arranges the text in the correct format for an ontology learning system and, generally, includes noisy text analytics. The extraction of terms begins with the tokenization or part of speech tagging to break texts into smaller constituents. In this phase, statistical or probabilistic measures are adopted for determining the “unithood”, the collocational stability of a noun sequence, and the termhood, the relevance or the specificity of a term with respect to a domain. Starting from the terms is possible to derive the concepts that can be formed by grouping similar terms and labeling them. The grouping phase involves discovering the variants of a term and grouping them together, while the concept’s label can be inferred by the use of existing background knowledge, such as WordNet, that may be used to find the name of the nearest common ancestor.

The relations model the interactions among the concepts in ontology: in general, two types of relations can be recognized in ontology: taxonomic and non-taxonomic relations. Taxonomic relations, that are hypernym, build hierarchies and can be labeled as “is-a” relations [8]. This kind of relations can be performed in various ways such as using predefined relations from existing background knowledge, using statistical subsumption models, relying on semantic similarity between concepts and utilizing linguistic and logical rules or patterns. The non-taxonomic relations are the interactions among the concepts other than hypernymy and their extraction is a challenging task. In this context verbs play a significant role such as the support of domain experts.

Axioms are propositions or sentences that are always taken as true and are the starting point for deducing other truth, verifying the correctness of the ontological elements and defining constraints. The process of learning axioms is still complex and there are few examples in the literature.

Ontology Learning has adopted well known techniques coming from different fields such as information retrieval, machine learning, and natural language processing [9]. This techniques can be generally classified into *statistics-based*, *linguistic-based*, *logic-based* or *hybrid* [28].

The statistics-based techniques are derived from information retrieval, machine learning, data mining and work at a syntactical level. In particular, these approaches are effective in the early stages of ontology learning, such as term extraction and hierarchy construction [4]. Some of the common techniques include clustering [27], Latent Semantic Analysis [25], term subsumption [17] and contrastive analysis [26]. The main idea behind these techniques is that the co-occurrence of words provides a reliable estimate about their semantic identity. In this way a concepts can be inferred.

The linguistics-based techniques can support all tasks in ontology learning and are based on natural language processing tools. In general some of the techniques include Part of Speech (POS) tagging, such as [2], syntactic structure analysis [19] and dependency analysis [7]. Other adopted techniques are related to semantic lexicon [22], lexico-syntactic patterns [5][24], subcategorization frames [18] and seed words [30].

The logic-based techniques and resources are the least common in ontology learning and are mainly adopted for more complex tasks involving relations and axioms. The two main techniques employed are inductive logic programming [20] and logical inference [23]. In the inductive logic programming, rules are derived from existing collection of concepts and relations which are divided into positive and negative examples. In logical inference, implicit relations are derived from existing ones using rules (transitivity and inheritance). In general it is difficult to say what of these techniques is the better one and, maybe, none of them is the only solution for the ontology learning.

As previously said, each phase of the ontology learning process can adopt one of these approaches in order to maximize the process effectiveness. In particular the terms and concepts extraction can be performed by the use of the statistics-based techniques while the inference of relations can be obtained by the use of linguistic and logic based techniques. In reality, the hybrid approach is mainly used in the existing studies and furnishes the best results [28][9].

Differently from the other described papers in ontology learning and population, that usually produces concept hierarchies by means of statistical and/or probabilistic methods (LSA, LDA, pLSA and so on), we enrich our terminological ontologies with the semantic features presented in general purpose lexical ontologies, such as WordNet. The use of both statistical and semantic techniques allows to have suitable and effective domain ontologies particularly suitable for a number of applications such as topic detection and tracking, opinion and sentiment analysis, text mining and classification [11], [15], [12].

The paper is organized as follows. Section 2 describes at glance our Ontology Learning and Population system architecture. Section 3 is devoted to a general description of the adopted methods and algorithms. Experiments and conclusions are reported in section IV and V respectively.

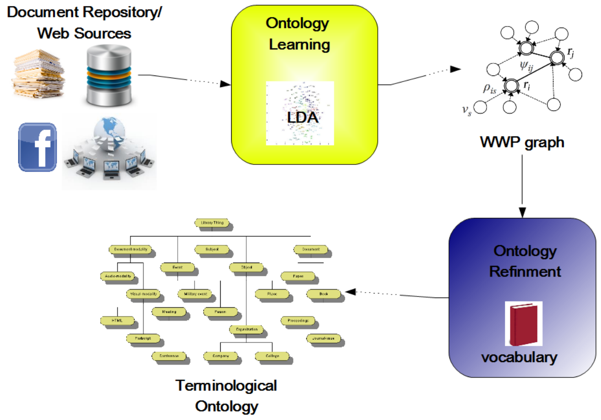


Fig. 1. The System Architecture.

II. System Architecture

Figure 1 describes the proposed system architecture. The system analyzes a number of documents coming from different web sources or from document collections related to a given domain of interests classified into a set of semantically disjoint topics. The system is formed by two main components:

- *Ontology Learning* Component, that uses *Latent Dirichlet Analysi*s (LDA) on the input documents and produces a *Weighted Word Pairs* (WWP) representation containing the most relevant domain concepts and their co-occurrence values (relations) in the analyzed set.
- *Ontology Refinement* Component, that using general purpose or domain-specific lexical databases, refines the previous discovered concepts, exploiting their lexical relationships (e.g. *is_a* taxonomic relations), adding hidden concepts, and producing the final ontology schema and population.

In the following we will discuss into details the basic components of the proposed architecture.

III. Terminological Ontology Building

A. Ontology Learning: concepts and relation extraction

In this section we explain how a WWP structure (Weighted Word Pairs) can be extracted from a corpus of documents.

The Feature Extraction module (FE) is represented in Fig.2. The input of the system is the set of documents:

$$\Omega_r = (\mathbf{d}_1, \cdots, \mathbf{d}_M)$$

After the pre-processing phase, which involves tokenization, stopwords filtering and stemming, a Term-Document Matrix is built to feed the Latent Dirichlet Allocation (LDA) [3] module. The LDA algorithm, assuming that each document is a mixture of a small number of latent topics and each word's creation is attributable to one of the document's topics, provides as output two matrices - Θ and Φ - which express probabilistic relations between topic-document and word-topic respectively. Under particular assumptions [13], [10], [14], LDA module's results can be used to determine: the probability for each word v_i to occur in the corpus $W_A = \{P(v_i)\}$; the conditional probability between word pairs $W_C = \{P(v_i|v_s)\}$; the joint probability between word pairs $W_J = \{P(v_i, v_j)\}$. Details on LDA and probability computation are discussed in [3], [16], [13].
Defining *Aggregate roots* (AR) as the words whose occurrence is most implied by the occurrence of other words of the corpus, a set of H aggregate roots $\mathbf{r} = (r_1, \dots, r_H)$ can be determined from W_C :

$$r_i = \text{argmax}_{v_i} \prod_{j \neq i} P(v_i|v_j) \qquad (1)$$

This phase is referred as Root Selection (RS) in Fig.2. A weight ψ_{ij} can be defined as a degree of probabilistic correlation between AR pairs: $\psi_{ij} = P(r_i, r_j)$. We define an *aggregate* as a word v_s having a high probabilistic dependency with an aggregate root r_i. Such a dependency can be expressed through the probabilistic weight $\rho_{is} = P(r_i|v_s)$. Therefore, for each aggregate root, a set of aggregates can be selected according to higher ρ_{is} values. As a result of the Root-Word level selection (RWL), an initial WWP structure, composed by H aggregate roots (R_l) linked to all possible aggregates (W_l), is obtained. An optimization phase allows to neglect weakly related pairs according to a fitness function discussed in [13]. Our algorithm, given the number of aggregate roots H and the desired max number of pairs as constraints, chooses the best parameter settings τ and $\boldsymbol{\mu} = (\mu_1, \dots, \mu_H)$ defined as follows:

1) τ: the threshold that establishes the number of *aggregate root/aggregate root* pairs. A relationship between the aggregate root v_i and aggregate root r_j is relevant if $\psi_{ij} \geq \tau$.
2) μ_i: the threshold that establishes, for each aggregate root i, the number of *aggregate root/word* pairs. A relationship between the word v_s and the aggregate root r_i is relevant if $\rho_{is} \geq \mu_i$.

Note that a WWP structure can be suitably represented as a *graph* $\mathbf{g}$ of terms (Fig. 3). Such a graph is made of several clusters, each containing a set of words v_s (*aggregates*) related to an *aggregate root* (r_i), the centroid of the cluster.

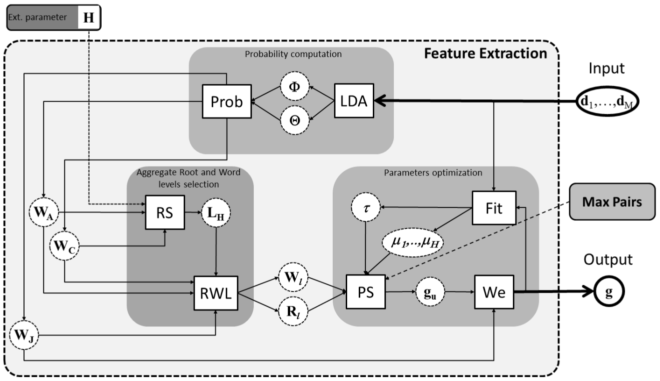


Fig. 2. Proposed feature extraction method. A *WWP* $\mathbf{g}$ structure is extracted from a corpus of training documents.

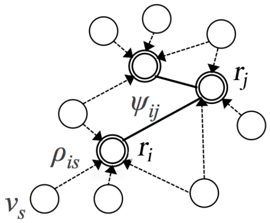


Fig. 3. Graphical representation of a *WWP* structure.

Aggregate roots can be also linked together building a centroids subgraph.

B. Ontology Refinement

The main goal of such a Component is to transform, for each topic, the WWP graphs into a terminological ontology, in order to represent and manage the knowledge coming from the document corpus in a more effective way. In the following, we will introduce some preliminary definitions and describe an algorithm for automatically building the described ontologies.

We first introduce the concept of a *Semantic Node* (w), as a triple w=$\langle s, REL, t\rangle$, s being the code of a given vocabulary synset, REL is the set of references to the other nodes and t the related concept label. A *Local Terminological Ontology* is a particular graph data structure $T =< w^*, W >$, w^* being the aggregate root node and W the set of the other semantic nodes. A *Domain Terminological Ontology* is a particular graph data structure $T =< W^*, W >$, W^* being set of the aggregate root nodes for a given semantic domain/topic and W the set of the other semantic nodes.

If we consider only *is_a* relationships among concepts, each semantic node is characterized by a single IS_A reference to the ancestor node and vice-versa, while a terminological ontology corresponds to a concepts taxonomy.

Algorithm 1 allows to build the local terminological ontology constituted by the taxonomy of discovered concepts for a single topic and for a given aggregate root node and by a set of generic relationships among the root node and other semantic nodes, using *WordNet* as general lexical vocabulary. The algorithm has in input the WWP graph and in particular considers the aggregate root node and a set of aggregated words (we also consider words containing the aggregated words) for a given topic/domain. In a first phase, the common hypernyms between the aggregate root node and aggregated words are determined and eventually added to the ontology as semantic nodes if they are semantically similar to the root node. In a second phase, ontology is updated by computing the correct IS_A relationships among the concepts, corresponding to the ancestor and leave nodes.

The following functions are exploited by the algorithm.

- **find_synset(**t**,**WN**)** - returns all possible WordNet synsets for a generic word t.
- **add_vector(**v_t^1**,**v_t^2**)** - adds a list of words to a vector of words.
- **find_composite_words(**t**,**WN**)** - returns the set of the words that contain a given word t in the WordNet database.
- **find_minimum_common_hypernym(**s**,**S**,**WN**)** - returns the minimum common ancestor with the related synset between a sysnset s and a set of synset S in the WordNet hierarchy.
- **add_ontology(**w**,**T**)** - adds a new node w to the taxonomy T and links the node with the related ancestor.
- **exists(**w**,**T**)** - returns true if a node with the same synset is already contained in the ontology T.
- **collapse(**v_t**,**WN**)** - modifies a vector of words collapsing the synonymous words in a unique term.
- **leaves_ontology(**T**)** - returns the leaf words in the ontology T.
- **ancestors_ontology(**T**)** - returns the ancestor words in the ontology T.
- **is_parent(**s_i**,**s_j**,**WN**)** - returns trues if the synset s_j is an ancestor of sysnet s_i in the WordNet hieararchy.
- **update_ontology(**w_{old}**,**w_{new}**,**T**)** - updates a node in the ontology T.
- **ordering (**v**,**WN**)** - performs and ordering of a vector of words on the base of the depth in the WordNet hierarchy.
- **semantic_distance (**s_i**,**s_j**,**D**)** - computes a semantic distance between two synsets using the *Wu & Palmer* metric [29] based on a domain specific dictionary D.

Eventually, generic relationships, whose semantics cannot be retrieved as IS_A relation in the WordNet vocabulary, are instantiated between root and aggregated nodes that appear in the concepts' taxonomy.

The previous algorithm is then iteratively repeated for each aggregate word in the considered domain and the obtained local ontologies are opportunely aligned and merged in a single domain terminological ontology, exploiting ontology-mapping techniques [6].

Figure 4 reports an example of terminological ontology (i.e., taxonomy of concepts with the related WordNet Synsets)

Algorithm 1 Local Terminological Ontology Building

```
Input: $v_t = [t_1, t_2, \ldots, t_n]$, a vector of aggregated words; $\langle \hat{t}, \hat{s} \rangle$, the considered aggregate root node with the related synset $\hat{s}$, $\gamma$ a given threshold.
Output: $T$, a Terminological Ontology.
Computing of the composite words that contain the input words
for $k = 1 \to n$ do
  $\tilde{v}_t$=find_composite_words($v_t[k]$,$WN$);
  add_vector($v_t$,$\tilde{v}_t$);
end for
Computing of the common ancestors
for $i = 1 \to m$ do
  $v_s$=find_synset($v_t[i]$);
  $\langle t^*, s^* \rangle$= find_minimum_common_hypernym($v_s[i]$, $\hat{s}$,$WN$);
  if ($t^* <> Entity \wedge$ semantic_distance($s^*$,$\hat{s}$)$\leq \gamma$) then
    $w^*$=$\langle s^*, NULL, t^* \rangle$;
    if (!exists($w^*$,$T$)) then
      add_ontology($w^*$,$T$);
    end if
    $\hat{w}$=$\langle \hat{s}, rel_{IS_A}(w^*), \hat{t} \rangle$;
    if (!exists($\hat{w}$,$T$)) then
      add_ontology($\hat{w}$,$T$);
    end if
    $w_t$=$\langle v_s[i], rel_{IS_A}(w^*), vt[i] \rangle$;
    if (!exists($w_t$,$T$)) then
      add_ontology($w_t$,$T$);
    end if
  end if
end for
Updating of the ontology
$v_w^a$=ancestors_ontology($T$);
ordering($v_w^a$,$WN$);
for $i = 1 \to length(v_w^a) - 1$ do
  for $j = i + 1 \to length(v_w^a)$ do
    if (is_parent($v_w^a[i].s$, $v_w^a[j].s$,$WN$)) then
      update_ontology($v_w^a[i]$, $\langle v_w^a[i].s, rel_{IS_A}(v_w^a[j]), v_w^a[i].t \rangle$,$T$);
    end if
  end for
end for
$v_w^l$=leaves_ontology($T$);
ordering($v_w^l$,$WN$);
for $i = 1 \to length(v_w^l) - 1$ do
  for $j = i + 1 \to length(v_w^l)$ do
    if (is_parent($v_w^l[i].s$, $v_w^l[j].s$,$WN$)) then
      update_ontology($v_w^l[i]$, $\langle v_w^l[i].s, rel_{IS_A}(v_w^l[j]), v_w^l[i].t \rangle$,$T$);
    end if
  end for
end for
```

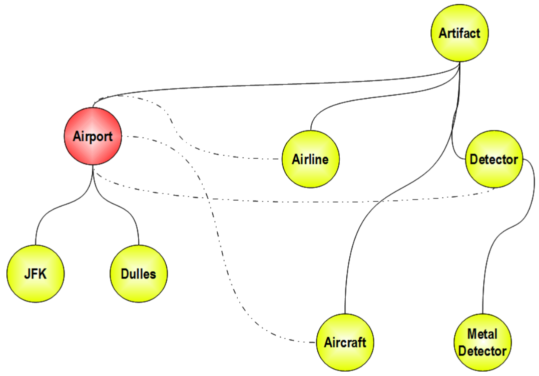


Fig. 4. An example of terminological ontology.

obtained by the algorithm application considering as root concept *airport* and as aggregated concepts *metal detector*, *airline*, *JFK*, *Dulles*. The relationships marked with broken line are generic relations, the other ones correspond to IS_A relationships.

IV. Preliminary Experimental Results

Evaluating the "quality" of an ontology is an important issue both for ontology delevopers and for final users: this task allows to compare different ontologies describing the same domain in order to choose the more suitable one for a given application.
The dataset from TREC-8 [1] collections (minus the Congressional Record) was used for performance evaluation. It contains about 520,000 news documents on 50 topics (no.401-450) and relevance judgements for the topics. Word stopping and word stemming with single keyword indexing were performed before building WWP.
In particular, we have selected for the experiments a subset of TREC documents related to the following topics:

- **osteoporosis** (21 documents),
- **cosmic events** (18 documents),
- **tropical storms** (118 documents),
- **airport security** (123 documents),
- **heroic acts** (94 documents),
- **robotic technology** (130 documents),
- **UV damage, eyes** (50 documents),
- **creativity** (75 documents),
- **counterfeiting money** (162 documents),
- **drugs, Golden Triangle** (136 documents).

For each topic, our system produced a terminological ontology: we have thus asked a set of users' groups to generate ontologies containing the concepts returned by the *Concept Extraction* component. The users have been grouped on the base of their expertise in the domain topics (namely high, low and medium experts). Each group was formed by about 10 students, for a total of about 200 users involved in the experimental process.

In particular, we have evaluated the *effectiveness* of our generated terminological ontologies with respect to humans on the base of the following several criteria: *Class Match Measure* (*CMM*), *DEnsity measure* (*DEM*), *Semantic Similarity Measure* (*SMM*), *BEtweenness Measure* (*BEM*) as in [1].

The *Class Match Measure* is meant to evaluate the "coverage" of an ontology for the given search terms. This measure evaluates class instances (high-level nodes in our case) in each ontology having labels matching a search term either exactly (node label identical to search term) or partially (node label contains the search term). An ontology that contains all search terms will obviously score higher than others, and exact matches are considered better than partial matches. The CMM can be obtained by the following equation:

$$CMM(\mathcal{O},T) = \alpha \cdot \sum_{c \in C(\mathcal{O})} \sum_{t \in T} I(c,t) + \beta \cdot \sum_{c \in C(\mathcal{O})} \sum_{t \in T} J(c,t) \tag{2}$$

where: $\mathcal{O}$ is the assigned ontology, $C(\mathcal{O})$ is the set of the high level nodes, T is the set of search terms, and $I(c,t)$ and $J(c,t)$ are two binary functions that return 1 in the case of a generic concept c of the ontology matching or containing a search term t respectively, 0 otherwise. For what the CMM metric computation concerns, for each topic we used as search terms the query keywords provided by TREC.

The *DEnsity Measure* is a metric that tries to measure the "representational density" or "informative content" of classes and consequently the level of knowledge detail. Density calculations are currently limited to the numbers of relations, subclasses, superclasses, and siblings for the different high-level nodes. We dropped the number of instances from this measure as this might skew the results unfairly towards populated ontologies which may not necessarily reflect the quality of the schema. The DEM can be obtained by the following equation:

$$DEM(\mathcal{O}) = \frac{1}{n} \cdot \sum_{i=1}^{n} \frac{rel(c_i) + sup(c_i) + sub(c_i) + sibl(c_i)}{M} \tag{3}$$

where: $\mathcal{O}$ is the assigned ontology, n is the number of matched high-level nodes respect to the search terms in the ontology, M is a normalization factor, $rel(c)$, $sup(c)$, $sub(c)$, $sibl(c)$ are apposite functions returning the number of relations, subclasses, superclasses, and siblings, respectively, of a generic concepts c_i.

The *Semantic Similarity Measure* calculates how close the classes that matches the search terms are in an ontology. The motivation for this is that ontologies whose position concepts further away from each other are less likely to represent the knowledge in a coherent and compact manner. The SSM formula used here is based on the *Rada Shortest Path* measure. SSM is measured from the minimum number

[1] http://trec.nist.gov/

of links that connects a pair of concepts. The SMM can be obtained by the following equation:

$$SSM(\mathcal{O}) = \frac{1}{m} \cdot \sum_{c, c_j \in C(\mathcal{O})} sim(c_i, c_j) \qquad (4)$$

where: $\mathcal{O}$ is the assigned ontology, m is the number of matchings for matched classes respect to the search terms in the ontology, $sim(c_i, c_j)$ is an apposite function returning the similarity (computing by the Rada measure) between two connected concepts c_i, c_j.

The *BEtweenness Measure* calculates the number of the shortest paths that pass through each couple of matched high-level nodes (*betweenness*) in the ontology. The nodes occurring on many shortest paths among other nodes have higher betweenness value than others. The assumption is that if a class instance has a high betweenness value in an ontology then this node is central to that ontology. Ontologies where those classes are more central will receive a higher score. The BEM can be obtained by the following equation:

$$BEM(\mathcal{O}) = \frac{1}{n} \cdot \sum_{k=1}^{n} \sum_{c_i \neq c_j} \frac{\sigma_{c_i c_j}(c_k)}{\sigma_{c_i c_j}} \qquad (5)$$

where: $\mathcal{O}$ is the assigned ontology, n the number of matched high-level nodes respect to the search terms in the ontology, $\sigma_{c_i c_j}$ and $\sigma_{c_i c_j}(c_k)$ are apposite functions returning the shortest path from c_i to c_j and the number of shortest paths from c_i to c_j that passes through a generic concept c_k respectively.

As we can see from Figure 5, our ontology has a quality index very close to that of an ontology generated by experts on the considered domain.

Finally, we measured the times of building a terminological ontology depending on the number of input documents for each topic [2]. We observed that the *drugs, Golden Triangle* ontology, built from 136 documents, requires less than 30 seconds for its complete building, thus ensuring enough scalability on more large data set.

V. Conclusions and future work

In this paper, we presented an ontology learning and population technique that exploit both statistical and semantic methodologies for generating terminological ontologies for a given semantic domain. The reported experimental results demonstrated the effectiveness of the proposed system in terms of goodness and quality of produced ontologies with respect to the ones manually generated by experts or less humans on the considered domain. Thus, the automatic generated ontologies can be suitably used topic for different application such as semantic-based retrieval, topic detection and tracking, sentiment analysis and so on.

[2] we use a Linux Ubuntu platform running on a 8GB RAM single CPU

References

[1] Harith Alani and Christopher Brewster. Metrics for ranking ontologies. In *4th Int. EON Workshop, 15th Int. World Wide Web Conf.*, 2006.

[2] Steven Bird, Ewan Klein, Edward Loper, and Jason Baldridge. Multidisciplinary instruction with the natural language toolkit. In *Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics*, TeachCL '08, pages 62–70, 2008.

[3] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.

[4] Christopher Brewster, Simon Jupp, Joanne Luciano, David Shotton, Robert Stevens, and Ziqi Zhang. Issues in learning an ontology from text. *BMC Bioinformatics*, 10(Suppl 5):S1, 2009.

[5] Paul Buitelaar and Bernardo Magnini. Ontology learning from text: An overview. In *In Paul Buitelaar, P., Cimiano, P., Magnini B. (Eds.), Ontology Learning from Text: Methods, Applications and Evaluation*, pages 3–12. IOS Press, 2005.

[6] Namyoun Choi, Il-Yeol Song, and Hyoil Han. A survey on ontology mapping. *SIGMOD Rec.*, 35(3):34–41, September 2006.

[7] Massimiliano Ciaramita, Aldo Gangemi, Esther Ratsch, Jasmin Šaric, and Isabel Rojas. Unsupervised learning of semantic relations between concepts of a molecular biology ontology. In *Proceedings of the 19th international joint conference on Artificial intelligence*, IJCAI'05, pages 659–664, 2005.

[8] Philipp Cimiano, Aleksander Pivk, Lars Schmidt-Thieme, and Steffen Staab. Learning taxonomic relations from heterogeneous evidence.

[9] Philipp Cimiano, Johanna Vlker, and Rudi Studer. Ontologies on demand a description of the state-of-the-art, applications, challenges and trends for ontology learning from text, 2006.

[10] F. Clarizia, F. Colace, M. De Santo, L. Greco, and P. Napoletano. Mixed graph of terms for query expansion. In *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*, pages 581–586, 2011.

[11] F. Clarizia, F. Colace, M. De Santo, L. Greco, and P. Napoletano. A new text classification technique using small training sets. In *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*, pages 1038–1043, Nov 2011.

[12] Francesco Colace, Massimo De Santo, and Luca Greco. A probabilistic approach to tweets' sentiment classification. In *Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on*, pages 37–42, 2013.

[13] Francesco Colace, Massimo De Santo, and Luca Greco. Weighted word pairs for text retrieval. In *Proceedings of the 3nd Italian Information Retrieval (IIR), volume 964 of CEUR Workshop Proceedings*, 2013.

[14] Francesco Colace, Massimo Santo, Luca Greco, and Paolo Napoletano. Improving text retrieval accuracy by using a minimal relevance feedback. In Ana Fred, JanL.G. Dietz, Kecheng Liu, and Joaquim Filipe, editors, *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, volume 348 of *Communications in Computer and Information Science*, pages 126–140. Springer Berlin Heidelberg, 2013.

[15] Francesco Colace, Massimo De Santo, and Luca Greco. An adaptive product configurator based on slow intelligence approach. *International Journal of Metadata, Semantics and Ontologies*, 9(2):128–137, 01 2014.

[16] Francesco Colace, Massimo De Santo, Luca Greco, and Paolo Napoletano. Text classification using a few labeled examples. *Computers in Human Behavior*, 30:689–697, 2014.

[17] Hermine Njike Fotzo and Patrick Gallinari. Learning generalization/specialization relations between concepts application for automatically building thematic document hierarchies.

[18] Pablo Gamallo, Alexandre Agustini, and Gabriel P. Lopes. Learning subcategorisation information to model a grammar with ”co-restrictions”, 2003.

[19] Andrew Hippisley, David Cheng, and Khurshid Ahmad. The head-modifier principle and multilingual term extraction. *Nat. Lang. Eng.*, 11(2):129–157, June 2005.

[20] Nada Lavrac and Saso Dzeroski. New York.

[21] Alexander Maedche and Steffen Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2):72–79, March 2001.

[22] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. Wordnet::similarity: measuring the relatedness of concepts. In *Demonstration Papers at HLT-NAACL 2004*, HLT-NAACL–Demonstrations '04, pages 38–41, 2004.

[23] Mehrnoush Shamsfard and Ahmad Abdollahzadeh Barforoush. The state of the art in ontology learning: a framework for comparison. *Knowl. Eng. Rev.*, 18(4):293–316, December 2003.

[24] Rion Snow. Semantic taxonomy induction from heterogenous evidence. In *In Proceedings of COLING/ACL 2006*, pages 801–808, 2006.

[25] Peter D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *Proceedings of the 12th European Conference on Machine Learning*, EMCL '01, pages 491–502, 2001.

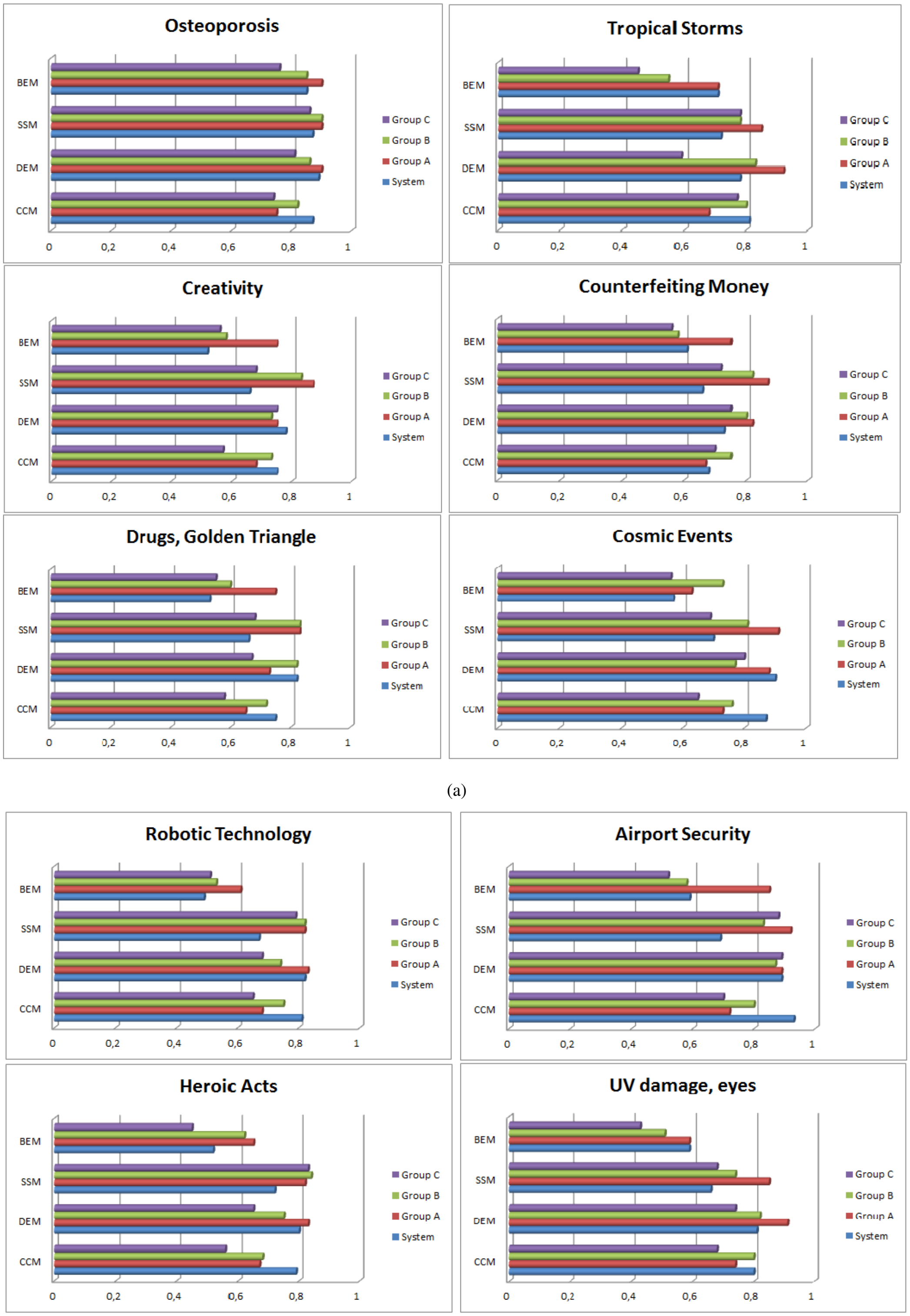
[26] Paola Velardi, Roberto Navigli, Alessandro Cucchiarelli, and Francesca Neri. Evaluation of ontolearn, a methodology for automatic learning of ontologies. In Paul Buitelaar, Philipp Cimiano, and Bernardo Magnini, editors, *Ontology Learning from Text: Methods, Evaluation and Applications*, pages 92–105. IOS Press.

[27] Wilson Wong, Wei Liu, and Mohammed Bennamoun. Tree-traversing ant algorithm for term clustering based on featureless similarities. *Data Min. Knowl. Discov.*, 15(3):349–381, December 2007.

[28] Wilson Wong, Wei Liu, and Mohammed Bennamoun. Ontology learning from text: A look back and into the future. *ACM Comput. Surv.*, 44(4):20:1–20:36, September 2012.

[29] Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. In *32nd. Annual Meeting of the Association for Computational Linguistics*, pages 133 –138, New Mexico State University, Las Cruces, New Mexico, 1994.

[30] Roman Yangarber, Ralph Grishman, and Pasi Tapanainen. Automatic acquisition of domain knowledge for information extraction. In *In Proceedings of the 18th International Conference on Computational Linguistics*, pages 940–946, 2000.



(a)

(b)

Fig. 5. Experimental Results.