A Name-Matching Algorithm for Supporting Ontology Enrichment

Alexandros G. Valarakos[1,2], Georgios Paliouras[1], Vangelis Karkaletsis[1], and George Vouros[2]

[1] Software and Knowledge Engineering, Laboratory
Institute of Informatics and Telecommunications,
National Centre for Scientific Research "Demokritos",
153 10 Ag. Paraskevi, Athens, Greece
{alexv, paliourg, vangelis}@iit.demokritos.gr
[2] Department of Information and Telecommunication Systems Engineering,
School of Sciences, University of the Aegean,
83200, Karlovassi, Samos, Greece
georgev@aegean.gr

Abstract. Ontologies are widely used for capturing and organizing knowledge of a particular domain of interest. This knowledge is usually evolvable and therefore an ontology maintenance process is required. In the context of ontology maintenance we tackle the problem that arises when an instance/individual is written differently (grammatically, orthographically, lexicographically), while representing the same entity/concept. This type of knowledge is captured into a semantic relationship and constitutes valuable information for many intelligent methods and systems. We enrich a domain ontology with instances that participate in this type of relationship, using a novel name matching method based on machine learning. We also show how the proposed method can support the discovery of new entities/concepts to be added to the ontology. Finally, we present experimental results for the enrichment of an ontology used in the multi-lingual information integration project CROSSMARC.

1 Introduction

Ontologies are becoming an essential component of knowledge-intensive methods and systems because of their potential to capture and represent domain knowledge in a machine understandable and processable format. According to the most cited definition [4] in the literature, an ontology is an explicit specification of a domain conceptualization. It denotes and organizes entities/concepts that exist in a domain of interest, using a formal declarative language. Ontologies provide a common basis of understanding through their structure and vocabulary, facilitating information/knowledge dissemination and reuse.

A domain ontology usually consists of concepts/entities/types of the domain which are captured and organized by various types of relationships that hold between them. Relationships which organize concepts hierarchically are called

vertical relationships (e.g. the 'is-a'relationship) in contrast to horizontal relationships (e.g. the 'synonymy' relationship) which link entities across the hierarchy. The instantiation of the concepts in a particular domain is performed by instances/objects/individuals that are members of concepts.

A domain ontology captures knowledge in a static way, as it is a snapshot of knowledge from a particular point of view that governs a certain domain of interest in a specific time-period. However, this knowledge changes according to the needs of the method/system that uses the ontology as well as the evolutionary tendency of the already captured knowledge. Ontology maintenance is defined as the task of adapting the captured knowledge to some specifications, performing the appropriate interventions in its structure and content. It is a difficult and expensive task as it requires the collaboration of both knowledge engineers and domain experts. Ontology learning can facilitate the maintenance process by using machine learning techniques to obtain knowledge from data.

One problem that is often encountered in many knowledge-intensive applications is related to the different appearance (orthographically, lexicographically) of an instance. For example, the processor name 'Pentium 2' can be written differently as 'Pentium II'or 'p2' or 'P II' or 'Intel Pentium 2' etc. The poor performance of many intelligent systems and methods, in particular information extraction, retrieval and integration systems, is due to their inability to handle such cases. A domain ontology can handle this by relating the appropriate instances through predefined relationships denoting this regularity. In the multilingual information integration project CROSSMARC[3], an ontology was used that defines a horizontal relationship named 'synonymy' to cope with this. The task of maintaining this aspect of the ontology is generally addressed by work on ontology enrichment, which aims at the extension and updating of the ontology avoiding duplicate instances.

In this article we present a method for enriching a domain ontology with instances participating in the relationship 'synonymy' of the CROSSMARC ontology. This work follows naturally from the work in [8] which deals with the enrichment of a multi-lingual ontology with new instances participating in the vertical 'instance-of' relationship. We drive a machine learning-based algorithm to identify new instances by training it on positive examples of instances. The algorithm that we present in this paper uses information compression principles, in order to classify the new instances into existing concepts, e.g. the new instance 'P2' belongs in concept 'Intel Pentium II', and to discover new concepts by grouping lexicographically similar instances. This task is known as name matching and in the context of natural language processing, it aims to detect orthographic co-references between named entities [1] in a text. Also, this task is encountered in the data base community, where its goal is to identify data records that describe the same object keeping the number of duplicates to a minimum [3] or performing normalization to the data.

[3] IST (IST 2000-25366) EU-founded project:
http://www.iit.demokritos.gr/skel/crossmarc

Section 2 describes the overall structure of the CROSSMARC ontology, whereas section 3 presents our method for ontology enrichment. Section 4, introduces the machine learning algorithm that performs name matching and in section 5 we expose the results of the conducted experiment. Finally, we conclude in section 6 in which we present also our plans for future work.

2 CROSSMARC Ontology

The main aim in the design of the CROSSMARC ontology was sufficient flexibility in order to secure: (a) customizability to different domains and languages and (b) quick maintainability by modifying only a limited set of features. For these reasons, the architecture of the ontology consists of four layers:

- The *meta-conceptual layer*, which defines the generic ontological commitments of the CROSSMARC ontology architecture. It includes three meta-elements: Feature, Attribute and Value. These are used in the conceptual layer to assign computational semantics to elements of the ontology.
- The *conceptual layer*, which comprises the concepts that populate the specific domain of interest. The internal representations of these concepts as well as their relations comply with the commitments defined in the meta-conceptual layer.
- The *instances layer*, which represents domain-specific individuals. Therefore, this layer instantiates each concept.
- The *lexical layer* provides the multi-lingual surface realization (lexicalization) of ontologies' concepts and instances in the natural languages that are being supported by the project, currently English, Greek, French and Italian.

The ontology that we used in our case study describes laptop products and has been manually constructed using the Protege-based [6] management system developed in the context of the CROSSMARC project. The ontology consists of 'part-of' relationships, which link the main concept, namely laptop, with its parts (e.g. processor, screen, battery, price etc.) Additionally, there is a 'has attribute' relationship for each concept which links them with other concepts (e.g. processor is linked with processor name), an 'instance-of' relationship that denotes the instances (members) of the concepts, e.g. 'Pentium 3' and 'amd k6' instantiates the concept 'processor name'. Furthermore, a 'synonymy[4]' relationship (non-taxonomic, horizontal relationship) links the appropriate different surface appearances - lexicalizations - of an entity that is classified to be an instance of a concept. All the above relationships are defined in the ontology's XML schema. The ontology consists of 119 instances for English and 116 for Greek. Also, instances are available for the Italian and French language. For the purposes of our case study, we use only the English instantiation of the ontology.

[4] The meaning of this word is overridden; it refers to the surface appearance of an instance rather to its meaning.

3 Name-Matching for Ontology Enrichment

3.1 Ontology enrichment

The ontology enrichment task is initiated when the knowledge captured in an ontology is out-of-date or incomplete for the task at hand. In our case study, we are interested in enriching the ontology with new instances, focusing on particular instances of the ontology that participate in the 'synonym' relationship. This is a strong relationship underlying the data of a domain and is part of many real-life domain ontologies. The laptop domain that we study here is a highly evolving one, as new laptop products appear in the market almost daily. Thus, the quick and accurate maintenance of the ontology is vital to the performance of the various modules [5] that use it, e.g. information extraction. For example, if the ontology is not aware of the new Intel's processor names and their different typographic appearances, the information extraction task will not be able present this information in a normalized form to the user.

The aim of our method is to discover different surface appearances of an instance by employing a learning algorithm on candidate instances that are produced by a well-defined methodology for ontology enrichment [9]. The overall methodology to ontology enrichment iterates through four stages:

1. Use the domain ontology to semantically annotate a domain-specific corpus.
2. Use the annotated corpus to train a Hidden Markov Model to locate new instances.
3. Extract new candidate instances from the corpus, using the trained Hidden Markov Model.
4. Use domain experts to validate the new instances and manually add them to the domain ontology.

The aim of the research proposed in this paper is to perform the fourth stage in a more robust, tolerant and therefore effective way, clustering different surface appearances of an instance via the synonym relationship. This will further reduce the involvement of domain experts in the whole process and ease the frequent update of the ontology with buzzwords appearing in publications.

3.2 Name Matching

The name matching task fits well with the problem of discovering instances that differ typographically (different surface appearance) but represent the same concept/entity. The task of matching entities has been researched by various communities, including statistics, databases, and artificial intelligence proposing different techniques. Cohen et. al [2] give a short survey on these techniques.

The novel algorithm that we use for name matching is based on the assumption that different lexicalizations of an entity use more-or-less a common set of 'core' characters. Therefore lexicalizations that are 'close' to this set are potential alternative appearances of the same entity, while those that are 'far' from this set are potentially related to another concept.

Specifically, the proposed algorithm classifies the candidate instances generated in the third stage of the ontology enrichment methodology, by employing a compression-based score function. The instances that already exist in the ontology, grouped by the concept they belong to, constitute the initial clusters. Each surface appearance of an instance is included only once in the cluster. Each of the initial clusters is coded by a codebook, which is defined by a Huffman tree created by the cluster's members. A new instance, with an unknown surface appearance, is assigned to a cluster if its addition to the codebook does not increase the size of the coded cluster by more than a pre-specified threshold. Otherwise, a new cluster is created which defines a new concept containing initially a single instance. It should be stressed that in this method we do not incorporate domain knowledge but we handle only strings as they appear in the text. Therefore, the proposed algorithm cannot compute clusters of lexicalizations of the same entity whose 'core characters' differ radically. Hence, synonymy relationships that are not based on surface appearance cannot be identified and homonymy ambiguities cannot be resolved.

4 Compression-based Name Matching

In this section we present the **COCLU** (**CO**mpression-based **CLU**stering) algorithm that we propose for the discovery of typographic similarities between strings (sequences of elements-letters) over an alphabet (ASCII or UTF character set), which are candidate instances for a domain ontology, the CROSSMARC ontology. It is a partition-based clustering algorithm which divides the data into several subsets and searches the space of possible subsets using a greedy heuristic. Each cluster is represented by a model, rather than by the collection of data assigned to it. This property classifies the algorithm to the conceptual or model-based learning algorithms. The cluster model is realized by a corresponding Huffman tree which is incrementally constructed, as the algorithm dynamically generates and updates the clusters by processing one string (instance's surface appearance) at a time.

The algorithm employs a new score function that measures the compactness and homogeneity of a cluster. This score function is termed herein *Cluster Code Difference (CCDiff)* and is defined as the difference of the summed length of the coded string tokens that are members of the cluster, and the length of the same cluster updated with the candidate string. This score function groups together strings that contain the same set of frequent characters according to the model of a cluster. A string/instance belongs in a particular cluster when its *CCDiff* is below a specific threshold and the smallest between the *CCDiff's* of the given string with all existing clusters. A new cluster is created if the candidate string cannot be assigned to any of the existing clusters. As a result, it is possible to use the algorithm even when no initial clusters are available.

Similar to many incremental algorithms, the order in which the strings are encountered influences the performance of the proposed algorithm. For this reason, when many candidate strings are available, as is usually the case in the

ontology enrichment process, we iteratively select the candidate instance that is more reliably assigned to a particular cluster. The instance that is selected in each iteration is the one that maximizes the difference between its two smallest *CCDiff's* from the existing clusters.

The algorithm implements a hill-climbing search in the subset space for locating the best partition. It iteratively computes the *CCDiff* for all the existing clusters and for all candidate strings/instances and selects the instance that can be more reliably assigned to a cluster. If the corresponding *CCDiff* is greater than a user-defined threshold then a new cluster is created. The pseudo-code of the COCLU algorithm is presented below:

```
0. Given CLUSTERS and candidate INSTANCES
1. while INSTANCES do
    a. for each instance in INSTANCES
         i.  compute  CCDiff for every cluster in CLUSTERS
    b. end for each
    c. select instance from INSTANCES that maximizes
        the difference between its two smallest CCDiff's
    d. if min(CCDiff) of instance > threshold
         i.   create new cluster
         ii.  assign instance to new cluster
         iii. remove instance from INSTANCES
         iv.  calculate code model for the new cluster
         v.   add new cluster to CLUSTERS
    e. else
         i.   assign instance to cluster of min(CCDiff)
         ii.  remove instance from INSTANCES
         iii. recalculate code model for the cluster
2. end while
```

If the CLUSTERS set is empty at the beginning, the algorithm chooses the longest string, in order to construct the first cluster. This part of the algorithm can be improved by defining a reasonable heuristic for choosing a "good" string to start with.

5 Experimental Results

We have evaluated the performance of the algorithm using the enrichment methodology presented in section 3 on the laptop domain ontology of CROSSMARC. The evaluation included two scenarios. In the first scenario, we evaluated the ability of the algorithm to discover new clusters (cluster generation), by hiding one or more of the existing clusters in each run. In the second scenario we evaluated the ability of the algorithm to assign a string/instance to the appropriate cluster, while decreasing proportionally the number of instances available initially in the cluster. In the first scenario we set the CCDiff threshold to 20 whereas in the second one high enough to avoid creating new clusters.

Table 1 presents some statistics about the clusters we used in our experiments. In this you can find the cluster's name and type, as well as the number of instances in it. The identifier attribute is used as reference to the cluster in the following sections.

Table 1. Statistical data about the clusters that were used

Cluster's Name	Cluster's Type	Number of Instances
Amd	Processor Name	19
Intel	Processor Name	8
Hewlett-Packard	Manufacturer Name	3
Fujitsu-Siemens	Manufacturer Name	5
Windows 98	Operating System	10
Windows 2000	Operating System	3

5.1 Concept Generation Scenario

In this experiment we hide incrementally one cluster at a time and measure the ability of the algorithm to discover the hidden clusters. A cluster is characterized by the majority of its instances. For example, a cluster which contains 6 instances of 'Windows 2000' and 3 of 'Windows 98' is characterized to be mostly a 'Windows 2000' cluster, rather than a 'Windows 98' cluster. We use three measures to evaluate the cluster generation process: 'correct' measures the clusters that were correctly generated according to the initial partition, 'wrong' measures the erroneously generated clusters and 'missed' measures the clusters that the algorithm did not manage to generate.

Initially, we conducted 6 different experiments, in which we hid a different cluster each time. The COCLU algorithm generated 2 wrong clusters while it succeeded not to miss any cluster. The two clusters that were wrongly generated correspond to instances of 5 and 6. In other words, the members of each of these two clusters were incorrectly further subdivided into two smaller clusters. The same phenomenon was observed in all of the experiments that we did, where we hid from 2 up to all 6 clusters. At each step, we enumerated all possible combinations of the 6 clusters and did the experiments. No missed clusters were observed in any experiment, while the 'wrong' clusters that were generated corresponded consistently to the subdivision of clusters 5 and 6. In standard information retrieval terms, the recall of the algorithm was 100%, as it managed to generate all the required clusters, while its precision was 75% in all of the experiments. The fact that the two figures remain unaffected by the number of hidden clusters indicates the robustness of the algorithm, which remains to be proven by more extended experimentation.

5.2 Instance Matching Scenario

In this experiment we measured the ability of the algorithm to assign an instance to the correct cluster (accuracy) reducing proportionally the instances in the initial clusters. By doing this, we tested the performance of our algorithm to handle cases where little background knowledge is used. Table 2 presents the results measuring the accuracy of the algorithm.

Table 2. Results of the instance matching experiments

Instance Reduction (%)	Correct	Accuracy (%)
90	3	100
80	11	100
70	15	100
60	19	100
50	23	95,6
40	29	96,5
30	34	94,1

As expected, the accuracy of the algorithm decreases as the size of the initial clusters is decreasing. However, it is very encouraging that, despite the small number of clusters, their size can be further reduced to almost half without any loss in accuracy. Additionally, the accuracy of the algorithm is preserved at a high level (above 90%) even when the size of the original clusters is reduced to less than a third of the initial.

6 Conclusions

We have presented a novel algorithm (COCLU) for the discovery of typographic similarities between strings facilitating in this way the enrichment of a domain ontology. We have integrated COCLU into a well-defined methodology for ontology enrichment in order to support a non-taxonomic relationship, namely 'synonym' (in the context of the CROSSMARC ontology), between instances. The algorithm performed very well obtaining remarkably good results both in terms of generating new clusters, as well as in assigning new strings to the correct clusters. Thus, the initial results are very encouraging, although further experimentation with larger and noisy datasets is needed, in order to prove the value of the method. Furthermore, the method will need to be compared experimentally with other similar methods that may be used to tackle the same problem.

In addition to the need for further experimentation, we are planning to improve and extend the algorithm in several ways. One issue that we are studying is a good heuristic for choosing a "good" string to start with when no cluster is given apriori. Furthermore, we are investigating the possibility of extending the algorithm beyond typographic clustering and into other interesting synonymy

relationships. In order to do this, we will need to take into account contextual information that will allow us to identify higher-order semantic relationships among the clusters.

References

1. Bontcheva, K., Dimitrov, M., Maynard, D., Tablan, V., Cunningham, H., Shallow Methods for Named Entity Co-reference Resolution, In Proceedings of TALN 2002, Nancy, 24-27 June 2002
2. Cohen, W., Ravikumar, P., Fienberg, S., A Comparison of String Distance Metrics for Name-Matching Tasks, In Proceedings of IIWeb Workshop, 2003
3. Galhardas, H., Florescu, D., Shasha, D., Simon, E., An extensible framework for data cleaning, In Proceedings of ICDE, 2000
4. Gruber, T. R., A translation approach to portable ontologies, Knowledge Acquisition, 5(2):199-220, 1993
5. Hachey, B., Grover, C., Karkaletsis, V., Valarakos, A., Pazienza, M. T., Vindigni, M., Cartier, E., Coch, J., Use of Ontologies for Cross-lingual Information Management in the Web, In Proceedings of the Ontologies and Information Extraction International Workshop held as part of the EUROLAN 2003, Romania, July 28 - August 8, 2003
6. Noy, N. F., Fergerson, R. W., MusenM. A., The knowledge model of Protege-2000: Combining interoperability and flexibility, In Proceedings of EKAW 2000, Juan-les-Pins, France, 2000
7. Pazienza, M. T., Stellato, A., Vindigni, M., Valarakos, A., Karkaletsis, V., Ontology Integration in a Multilingual e-Retail System, In Proceedings of the HCI International Conference, Volume 4, pp. 785-789, Heraklion, Crete, Greece, June 22-27 2003
8. Valarakos, A., Sigletos, G., Karkaletsis, V., Paliouras, G., Vouros, G., A Methodology for Enriching a Multi-Lingual Domain Ontology using Machine Learning, In Proceedings of the 6th ICGL workshop on Text Processing for Modern Greek: from Symbolic to Statistical Approaches, held as part of the 6th International Conference in Greek Linguistics, Rethymno, Crete, 20 September, 2003
9. Valarakos, A., Sigletos, G., Karkaletsis, V., Paliouras, G., A Methodology for Semantically Annotating a Corpus Using a Domain Ontology and Machine Learning, In Proceedings of the International Conference in Racent Advances in NLP (RANLP), Borovest, Bulgaria, 2003