LabelTranslator: A System for Localizing Ontologies in Collaborative Settings

Mauricio Espinoza\textsuperscript{a,c}, Asunción Gómez-Pérez\textsuperscript{a} and Eduardo Mena\textsuperscript{b}

\textsuperscript{a} Departamento de Inteligencia Artificial, UPM, 28660 Boadilla del Monte, Spain, E-mail: mespinoz@ucuenca.edu.ec, asun@ifi.upm.es
\textsuperscript{b} IIS Department, Univ. of Zaragoza, María de Luna 1, 50018 Zaragoza, Spain E-mail: emena@unizar.es
\textsuperscript{c} Facultad de Ingeniería, Universidad de Cuenca, Cdla. Universitaria Av. 12 de Abril s/n, Cuenca, Ecuador

Abstract. LabelTranslator is a system to semi-automatically localize ontologies in different natural languages: it extracts, translates and updates the lexicon of an ontology. Its objective is to support the development of multilingual ontologies, combining and exploiting the linguistic and semantic knowledge available on the Web to discover the translations of each ontology entity. We present the design of LabelTranslator, which have been guided by the requirements of international institutions as FAO for localizing different types of resources such as thesauri or ontologies. In particular, the three main activities handled by LabelTranslator are detailed: managing the multilingual content for localization, monitoring and control the localization activity, and discovering the more appropriate translations for each ontology entity.

Keywords: Knowledge Engineering, Ontology Localization, Collaborative Ontology Localization

1. Introduction

On the light of the growing interest and use of Semantic Web technologies, the need for ontology-based applications that manage and interact with different natural languages has also arisen. Many are the organizations operating at an international level that have to manage information in several natural languages, and want to offer their users the possibility of accessing and retrieving data in their own languages.

International institutions with information and resources in multiple languages, such as the FAO\textsuperscript{1} in the Agricultural domain or the WHO\textsuperscript{2} in the Health area, are currently working in the introduction of semantic technologies in their information systems. One of their main concerns is to account for multilinguality and the localized representation of knowledge [1].

Thus, multilinguality and localization become a real need in the current Semantic Web, and research in this sense has focused on the representation of linguistic and multilingual information in ontologies [2,16], and on the automatic localization of ontologies [6,7]. In order to be able to exploit ontologies in a multilingual settings, an adaptation of the ontology to different natural languages is required.

Ontology Localization has been defined as the process of adapting a given ontology to the needs of a certain community, which can be characterized by a common language, a common culture, or a certain geo-political environment [3]. The difficulties involved in the localization of ontologies have been discussed in [8], as well as the strategies that can be followed depending on the final aim of the localized ontology, and the representation modalities that better comply with each strategy. Additionally, guidelines have also been proposed for the localization of ontologies with the aim of helping users in this task.

Regarding ontology localization tools, LabelTranslator [6,7], implemented as a plugin of the NeOn Toolkit [9], supports a semi-automatic localization of ontologies to different natural languages, and in its cur-
rent version, it can localize ontologies in English, German, and Spanish. The main goal of LabelTranslator is to reduce the cost of translation and minimize the time to localize an ontology. The first prototype presented in [5] gave us an insight in the potential of this technology and it provided a single working scenario, in which only one person was in charge of supervising all the steps in the ontology localization activity. However, it lacked of functionalities to support collaboratively the localization of ontologies in those settings in which many actors geographically disperse, using diverse natural spoken languages where involved. Thus, the maintenance cycle of each language will often be carried out separately. To address these limitations, we created a workflow-based model for the collaborative localization of ontologies in distributed environments.

In this paper, our objective is to describe the approach we propose for the management of a collaborative ontology localization, and the components required to support it. Our contribution also includes the implementation of the proposed approach. The rest of the paper is structured as follows. In section 2 the main modules for supporting the localization of ontologies in distributed and collaborative environments are introduced. A description of the modules of our generic architecture is presented in sections 3, 4, and 5. Section 6 presents a set of experiments used to evaluate the technological aspects of the LabelTranslator system. Finally, conclusions and future work appear in section 7.

2. Architecture of Collaborative LabelTranslator

In this section we present a generic architecture for localizing ontologies in distributed and collaborative environments. Both Automated Translation and Collaboration and Distribution have been the key factors in the design of the advanced version of LabelTranslator. To increase the quality of the translations in automatic localization systems we propose a collaborative and distributed approach to: i) enhance the communication and collaboration among localization stakeholders, and ii) increase the participation of ontology users in the ontology localization process.

LabelTranslator has been designed around three core activities, each corresponding to a “layer” of its architecture (see Figure 1):

- The Ontology Management module enables ontology editors and managers to automatically access and manage multilingual content to be localized.
- The Collaborative Localization Management is the module designed to help manage, monitor, and control the localization activity itself.
- The Ontology Translator module is the core of the whole system. It allows to automatically discover the more appropriate translations for each ontology entity.

At a more technical level, these three modules are hosted on centralized client-server architecture in which the server maintains the current state and full history of all localized ontologies. The descriptions of the three main tasks of LabelTranslator, both at the conceptual and technical levels, are presented in the following sections.

3. The Ontology Management Module

Ontology Management is the module designed to help ontology engineers and managers to manage multilingual content for localization. The main functionalities of the Ontology Management component are: i) development and storage of ontologies that need to be localized, ii) selection of the ontology elements to be translated into different natural languages, iii) handling of different versions of the localized ontologies, and iv) deployment of all multilingual information associated with ontology elements.

Analyzing the state-of-the-art on ontology management tools, we observe that the evolution of semantic technologies has led to a number of concrete implementations to support specific ontology engineering activities. Basically, the first two functionalities (above described) are well supported by ontology management tools. However, popular tools for ontology development are limited with respect to how to model multilinguality in ontologies.

In our approach we have implemented the multilinguality support on NeOn Toolkit, a state-of-the-art, open source multi-platform ontology engineering environment, which provides comprehensive support for the ontology engineering life-cycle. We have extended the architecture of the NeOn Toolkit incorporating two components: the Localization GUI component and the Ontology Repository component.

Readers interested in the state-of-the-art of ontology management tools can refer to [15].
3.1. Localization GUI component

This component provides additional extension points to modify the main components of the NeOn Toolkit, with the aim of controlling the aspects related to the localization activity. For the different localization stakeholders, a whole set of views\(^4\) has been developed, which interacts with the different modules of the system. Along this paper we will show some of views used in our tool.

3.2. Ontology Repository component

This component captures the multilingual linguistic information associated with ontology elements. LabelTranslator supports the Linguistic Information Repository model (LIR) [16] designed for the representation of multilingual information in ontologies. The inclusion of the LIR in the system ensures the separation of information that is considered orthogonal in nature (we refer to the ontological and linguistic information). This way of representation is the current trend in the integration of multilinguality in ontologies [2].

The main advantages of the LIR model rely on its flexibility by allowing i) the enrichment of any ontology element with as much linguistic information as needed by the final application (lexicalizations, definitions, sources of provenance, etc.), and ii) the establishment of links among linguistic elements within and across different natural languages. Within the same language, lexical and terminological variants of the lexicalization of an ontology element can be accounted for. Across languages, the LIR also allows to define equivalence relationships between lexicalizations in different languages, as well as account for conceptualization mismatches. The rationale underlying the LIR is not to design a lexicon for each language involved in the localization activity, but to express the knowledge represented in the ontology so that it can be understood and reused by the target culture.

In the Figure 2 we show the Linguistic Information view implemented in LabelTranslator to allow ontology users to manage the multilingual information provided by the LIR model. The page shows five sections that correspond to the lexical entries of the selected ontology element (FAO in our example). For instance, in this case the concept FAO has three lexical entries, two in English and one in Spanish.

In our approach, the LIR model is represented as an ontology, whose instances represent the lexical knowledge. All information managed by the LIR model is controlled by a specialized unit of the ontology repository component. This unit provide the following features:

- It provides a special API to retrieve linguistic knowledge or to update the linguistic model. Also, it acts as a wrapper around any possible representation of the model.
- It implements both load and save mechanisms which can serialize the lexical entries associated with one ontology.

\(^4\)In the NeOn Toolkit a view is typically used to navigate a hierarchy of information, open an editor, or display properties for the active editor.
4. The Collaborative Localization Management Module

This section describes in detail the goal and functionalities of the Localization Management which is the key module that allows localization managers to monitor, manage and control the localization activity in collaborative and distributed settings. The Workflow Localization Manager (WLM) is the core component of this module. The main goals of the Collaborative Localization module are:

1. To manage the time flow of the localization activity from the start to the end,
2. To detect the changes in the Ontology Management module, and
3. To manage the individual localization tasks performed in the Translator module.

4.1. Workflow Localization Manager (WLM)

In order to support the first goal, the WLM includes a collaborative workflow that implements the necessary mechanisms to allow ontology stakeholders to perform the activities of the ontology localization activity. Thus, the collaborative workflow is responsible for the coordination of who (depending on the user role) can do what (i.e., what kind of actions) and when (depending on the status of the ontology elements).

From a technical point of view, the collaborative workflow is associated with a set of initialization parameters (e.g., user roles, assigned tasks, etc), source and target languages, and a partially ordered set of activities. The WLM stores individually the initialization parameters of each ontology. However, the information about user, roles and skills is stored in a shared database, which has two benefits:

1. Improved Project Staffing. The Localization Managers can see all the information related to a participant (e.g., language skills). This saves time and allows for better decisions when staffing a new ontology localization project.
2. Shared Information Across Ontology Projects. The shared information provides a particular benefit for ontology projects that need to localize several ontologies. Maintaining a single user database allows to share users in different ontology projects.
In order to configure the workflow localization parameters, the WLM module extends the NeOn toolkit with a set of wizards. For example the user wizard shown in Figure 3, allows managing the profile of each participant of the localization activity. The wizard records information about the skills of each participant (source and target languages), and it describes the roles, operations and policies that apply to a certain ontology. All this information is used in LabelTranslator to check the users credentials at login time, and to determine whether a user is allowed to perform a certain operation based on the policies of the ontology to be localized. In our approach a user can play several roles in the localization activity. For example, a given user could play the role of Translator and Reviewer.

Coming back to the description of the workflow, the activities supported are: selecting the ontology elements to be translated, translating the selected elements, reviewing the translations, and updating the ontology with the linguistic information obtained. These activities summarize the localization tasks commonly followed by different organizations such as FAO.

In the next sections we explain the rest of the associated components.

4.2. Synchronization component

Since our approach follows the current trend in the integration of multilinguality in ontologies, the Synchronization component monitors the changes in the ontology model and automatically propagates those changes to the linguistic model using synchronization techniques.

In order to keep both models, the ontology model (OM) and lexical model (LIR), synchronized, we first need to find out exactly what has been changed in the ontology model, then find the associated information in the linguistic model and, only then, start the updating. Basically, the addition of new terms in the ontology, or deletion of an existing term can be controlled by some mechanism of change. In the NeOn Toolkit, an advanced change tracking based on Resource Delta is able to capture changes even when ontological terms have changed their position within the ontology model. By adopting this feature, the synchronization component can accurately identify the minimal set of changes needed to adjust the structure of the linguistic model, a critical first step to ensure that a change is made properly in the localized ontology. To correctly update the linguistic model, the system identifies:

---

\[5\] A wizard is a user interface element that presents a user with a sequence of dialog boxes that lead the user through a series of well-defined steps. Tasks that are complex, infrequently performed, or unfamiliar may be easier to perform using a wizard.

\[6\] A resource delta represents changes in the state of a resource tree between two discrete points in time.
1. any ontology term in the original ontology whose label have changed in the updated ontology,
2. any ontology term that has been added to the updated ontology,
3. any ontology term which has been removed from the original ontology, and
4. any ontology term whose position in the updated ontology differs from that in the original ontology.

Finding where a translation is required is only part of the problem. We also need to ensure that changes in the ontology structure are accurately propagated to the linguistic information of each node in the distributed network that maintains a copy of the localized ontology (translators and reviewers). In our approach, the ontology elements whose structure needs to be updated are clearly flagged in each node. In this sense, the relevant structural changes are indicated in a form that turns updating the translation into a simple process, thus involving minimal work on the part of the translator or reviewer expert.

4.3. Localization Monitoring component

The last goal of the Localization Management module is supported by the Localization Monitoring component. This component controls and manages the activities that the ontology stakeholders are allowed to perform depending on their roles and the status of the ontology elements to be localized. The possible tasks in the collaborative workflow (described previously) apply at different levels of abstraction. In our solution we consider two levels: ontology element level and translation level. Regarding the ontology model, LabelTranslator deals with OWL-DL ontologies. Concepts, properties, instances and ontology term comments make up the set of ontology elements we are taking into account for the localization.

The possible states that can be assigned to ontology elements are:

- **In Use:** This is the status assigned to any ontology element when it passes first into the collaborative workflow, or when it was localized and then updated in the Ontology Repository.
- **New:** If the ontology element is added to the ontology once the ontology has been localized, the ontology element is passed to the “New” status, and remains there until it is localized.
- **Changed:** If the original label of the ontology element has changed, the element is passed to the “Changed” status, and remains there until the element is checked again due to localization needs.
- **Unused:** If the ontology element has been deleted, it is passed to the “Unused” status, and remains there until the ontology is synchronized (see synchronization component in the previous section).

The localization component controls also the status of the translations. Figure 4 shows the workflow of the translation level. States are denoted by rectangles and actions by arrows. The actors on the figure specify the actions that an ontology stakeholder can perform depending on their role. In the following we provide a detailed explanation:

- **Not translated:** This is the status assigned to any translation when it first passes into the collaborative workflow or when any change has been performed in the element of the ontology under consideration.
- **To be Translated:** Once the Localization Manager selects the translations with not translated status, these translations are passed to the “To be Translated” status, and remain there until a “Translator” translates them.
- **Auto translated:** If a “Translator” uses the automatic translation algorithm provided by the system, then translations are passed to the “Auto-translated” status, and remain there until the own “Translator” sends them to the “To Be Reviewed (automatic)” status.
- **Translated:** If a “Translator” discovers manually a translation, then the translation is passed to the “Translated” status, and remains there until the own “Translator” sends the translation to the “To be Reviewed” status.
- **To be Reviewed:** If a “Reviewer” approves the translations send by the translator, they pass to the “Complete” status. The reviewer knows in advance if translations have been discovered automatically or manually. For example, the word “automatic” in the message “To Be Reviewed (automatic)” indicates to reviewers that these translations have been obtained automatically. Additionally, when the translations reach the “Complete” status, they are automatically updated in the Ontology Repository.

In our implementation, the localization component performs automatically the actions defined in the workflow. Thus, this component takes care of enforcing the constraints imposed by the collaborative
workflow. In detail, whenever a new workflow action is performed, the component performs the following tasks: i) It gets the identity and role of the user performing the action, ii) It gets the status of the ontology element/translation associated to the action/change, iii) It verifies that the role associated to the user can perform the requested action when the ontology element/translation is in that particular status, iv) if the verification succeeds, it performs the workflow action (e.g., enabling all corresponding fields in the views); else neither action is performed.

Additionally, LabelTranslator extends some views in the Neon toolkit which allow ontology localization stakeholders: i) to see the appropriate information of the translations in the workflow, and ii) to perform the applicable workflow actions (select, translate, review, etc.), depending on their role.

Figure 5 shows the perspective\(^7\) used by LabelTranslator in order to support the localization workflow. The Ontology Navigator in the figure is located on the left side of the main view. It contains all ontologies that need be localized. The localization view is located on the middle of the main view. This view is used to add or to update the translations associated with the ontology terms that has been selected in the project tree on the left. Each ontology term is located in its own row. The localization view contains several shortcuts that make work faster, and are enabled according to user profile. Finally, the filter view is located on the right side of the figure. It contains several check box that allow user to modify what items are shown in the localization view.

\(^7\)A perspective is a visual container for a set of views and content editors.

5. The Translator Module

The goal of the Translator Module in LabelTranslator is to discover semi-automatically the more appropriate translations for each ontology entity. The main steps given by the translator module are: label pre-processing, label translation and label post-processing; (see Figure 6 which shows the steps presented in Figure 1 in more detail). With the help of the translator module, the translators/reviewers can reduce the effort and time spent to localize an ontology manually. In the rest of this section we will describe the main steps of the Translator module.
5.1. Label Pre-Processing

Ontology label pre-processing is essential in an ontology localization system, in order to simplify the core translation processing and make it both quality and time effective. The ontology labels pose different challenges to machine translation, which can be attributed to two distinct characteristics:

- Ontology labels differ linguistically and stylistically from written language: phrases are shorter and in some cases poorly structured, also they can contain ungrammaticality expressions (e.g., Service_Transport instead of Transport_Service)
- The most used approach for naming the ontology labels is to use a CamelCase approach. Therefore, we cannot rely on the initial uppercase letter to identify a phrase initial word or recognizing proper name, since names cannot be identified by an initial capital.

These problematic factors are dealt with a pre-processing pipeline that prepares the input for processing by a machine translation module. This approach has been used in a wider range of applications [21,20] where the task of the pre-processing pipeline is to make the input amenable to a linguistically-principled, domain independent treatment. In LabelTranslator, this task is accomplished in two ways:

1. By normalizing the input, i.e., removing noise, reducing the input to standard typographical conventions, and also restructuring and simplifying it, whenever this can be done in a reliable, meaning-preserving way.
2. By annotating the input with linguistic information, whenever this can be reliably done with a shallow linguistic analysis, to reduce input ambiguity and make a full linguistic analysis more manageable.

In the following we describe the functionalities of the different tasks in more detail:
5.1.1. Normalization

The label normalization groups two components, which clean up and tokenize the input. The text-level normalization phase performs operations at the string level (ontology term comments by example), such as removing strange text and punctuation (e.g., brackets, used to mark synonyms or usage context), or removing periods from abbreviations.

The tokenization phase breaks an ontology label into words. The token-level normalization recognizes and annotates tokens belonging to special categories (times, numbers, etc.), recognizes and identifies compound words. For annotating the tokens, LabelTranslator relies on Treetagger system [18]. The process of identifying of compound terms uses the Yahoo Term Extraction API.9

5.1.2. Tagging

In the tagging phase a tagger system10 assigns parts of speech to tokens. Part of speech information is used by the subsequent pre-processing modules, and also in parsing, to prioritize the most likely lexical assignments of ambiguous items. Treetagger is used in our approach.

5.1.3. Proper name recognition

Proper names are ubiquitous in ontology labels, specially in instance terms. Their recognition is important for deciding what instances should be translated, avoiding an annoying effect if any instance term is systematically mistranslated (e.g., a sport domain ontology where the golfer named Tiger Woods is an instance systematically referred to as “los bosques del tigre” (in Spanish), lit. “the woods of the tiger”).

Name recognition is harder in the ontology domain by the fact that capitalization information is used commonly for naming all type of ontological terms (concepts, properties, and instances), thus making unusable all methods that rely on capitalization as the main way to identify candidates. Of course, this problem is even more complex when no capitalization information is given. For instance, an expression like “mark shields”, as a possible instance in the ontology, is problematic in the absence of capitalization, as both ‘mark’ and ‘shields’ are three-way ambiguous (proper name, common noun, and verb). Our approach does not support the proper name recognition for the moment.

5.1.4. Segmentation

Segmentation breaks an ontology label into one or more segments, which are passed separately to subsequent modules. For our purposes, the translation units that we identify are syntactic units, motivated by cross-linguistic considerations. Each unit is a component that can be translated independently. Its translation is independent to the context in which the unit occurs, and the order of the units is preserved by translation.

The main motivation for segmenting are the problems found in the conventional online machine translation systems (e.g., GoogleTranslate, BabelFish, etc.). These systems have serious problem in dealing with long sentences due to the grammar coverage, memory limitation and computational complexity. Without proper treatment of long phrases, these systems, may fail to produce understandable translations. Although in our proposal we did not treat the translation of phrases (as the found in term annotations), we considered this component of utmost importance for future versions of the system.

In our approach we use a basic segmentation process to divide the tokens of a compound label. However, for the translation of phrases we devised a segmentation component based on machine learning techniques [12], syntactic analysis techniques [11], or support vector machines [13].

5.2. Ontology Label Translation

After preparing the ontology element for an effective machine translation processing, the Ontology Translator invokes the label translation component, which obtains the most probable translation for each ontology entity. This component integrates different machine translation approaches, combining the output by means of different translation combination strategies. The output of this component is a ranked set of translations for each ontology entity.

We have identified two translation strategies: one for simple labels and other for compound labels. These strategies are inspired in all empirical studies described in the literature about multi-engine machine translation architectures which operate by combining outputs from different translation engines. In the following we describe these translation strategies:

5.2.1. Translating simple labels

The strategy used for the translation of simple labels is a hybrid composition of two components, see also figure 7. The first component combines two differ-

9http://developer.yahoo.com/search/content/V1/termExtraction.html
10A tagger system is a tool for annotating text with part-of-speech and lemma information.
ent translation paradigms to discover candidate translations. The second component use an approach based on ontologies to discover the semantic senses of each translated label. In the figure 7, ovals represent the context extracted from lexicon and core ontology, and the translation results are represented as concentric circles.

First component - obtaining candidate translations

The first component takes as input an ontology label \( l \) described in a source language and returns a set of possible translations \( T = \{ t_1, t_2, \ldots, t_n \} \) in a target language. In order to discover the translations of each ontology label, each translation method accesses different lexical resources. On the one hand, the terminological-based approach use IATE\(^{11}\). On the other hand, the dictionary-based approach use the multilingual dictionary Wiktionary\(^{12}\). A buffer stores previously translations to avoid accessing the same data twice.

The algorithm used by the component is summarized in the following:

1. If the selected ontology label is already available in the target language in our buffer, then LabelTranslator just displays it, with all the relevant available information.
2. If the translation is not stored locally, then each translation method accesses remote repositories to retrieve possible translations. A simple disambiguation process based on term POS tagging is used to avoid a explosion of nuisance candidate translations.
3. If no results are obtained from the two previous steps, then the user can enter his/her own translation (together with the definition).

To combine the output of the different translation methods we use a linear combination. We assigned a major weight to obtained translations from the dictionary-based approach, because the translations obtained from these resources in our tests had high quality. In our approach, the translation of an ontology label denoted by \( t \), is a tuple \( \langle \text{trs}, \text{senses} \rangle \), where \( \text{trs} \) is translated label in the specific target language, and \( \text{senses} \) is a list of semantic senses extracted from different knowledge pools. In the following we briefly describe the task of automatically retrieving the possible semantic senses of a translated label (second component in our translation strategy).

Second component - obtaining semantic senses

In order to discover the senses of each translated label \( (t_i) \), we have considered the ontology-based approach proposed in a previous work \([19]\). Our system takes as input a list of words (each \( t_i \)), discovers its semantics in run-time and obtains a list of senses extracted from different ontology pools: Watson\(^{13}\), which indexes many ontologies available on the Web, remote lexical resources as EuroWordnet\(^{14}\), and other ontologies not indexed by Watson to find ontological terms that match those translated labels. We summarize here the key characteristic of the sense discovering process:

1. To discover the semantic of the input words, the system relies on a pool of ontologies instead of just a single ontology.
2. The system builds a sense (meaning) with the information retrieved from matching terms in the ontology pool.
3. Each sense is represented as a tuple \( s_k = <s, \text{grph}, \text{descr}> \), where \( s \) is the list of synonym names\(^{15}\) of keyword \( k \), \( \text{grph} \) describes the sense \( s_k \) by means of the hierarchical graph of hypernyms and hyponyms of synonym terms found in one or more ontologies, and \( \text{descr} \) is a description in natural language of such a sense if available.
4. As matching terms could be ontology classes, properties or individuals, three lists of possible senses are associated with each keyword \( k \): \( S_{\text{class}}^k, S_{\text{prop}}^k \), and \( S_{\text{indv}}^k \).
5. Each keyword sense is enhanced incrementally with the synonym senses (which also searches the ontology pool).
6. A sense alignment process integrates the keyword sense with those synonym senses representing the same semantics, and discards the synonym senses that do not enrich the keyword sense.

A detailed description of this process can be found in \([19]\). In order to perform cross-language sense translations, the external resources are limited to those resources that have multilingual information like EuroWordNet; however other resources can be used too. For example, a specific domain resource for the FAO...
(Food and Agricultural Organization) is Agrovoc\textsuperscript{16}, which could cover the vocabulary missed in EuroWordNet.

Once identified the semantic senses, the ontology-based method uses a ranking method to sort the list of translations according to similarity with the structural context of the label to be translated. The ranking method relies on the disambiguation algorithm described in [19]. Once all the translations are ranked, the method allows two operation modes:

- Semi-automatic mode: It shows a list with all the possible translations sorted decreasingly. The method proposes the most relevant translation to be selected first although the user can change this default selection.
- Automatic mode: It automatically selects the translation with the highest score.

5.2.2. Translating compound labels

Compound labels which have an entry in linguistic resources such as lexical databases, dictionaries, etc. (for example, “jet lag”, “travel agent”, and “bed and breakfast”) are treated as single words in our approach. Others like “railroad transportation”, which have no entry in the previous resources, are translated using a compositional method (see figure 8).

LabelTranslator uses a hybrid composition of two translation components for translating compound labels. In the following we describe both components.

First component - obtaining candidate translations

The first component is similar to the first component used for translating simple labels. However, in this case we have incorporated a method based on online machine translation systems in the parallel combination. The fundament behind of this approach is the vocabulary limitation (specially for compound labels) of both terminology and dictionary translation approaches. The online translation approach uses different multilingual systems such as GoogleTranslate\textsuperscript{17}, Babelfish\textsuperscript{18}, and FreeTranslation\textsuperscript{19}.

First, the component splits the label into tokens (“railroad” and “transportation” in the example); the individual components are translated and then combined into a compound label in the target language. Care is taken to combine the components respecting the word order of the target language (see second component below).

Second component - compositional method

The second component relies on a similar approach to the example-based machine translation techniques (EBMT) [17]. The main idea behind EBMT is that a given input phrase in the source language is compared with the example translations in the given bilingual parallel text to find the closest matching examples can be used in the translation of that input phrase.

One of the main approaches in the EBMT paradigm is to use pattern matching techniques. First, these approaches collect word sequences from each corpus by first using translation patterns to acquire candidates for bilingual expressions. Second, a search for pairs of words that satisfy the correspondences of the sequences is performed.

Two are the main differences in our approach. First, instead of extracting the bilingual translation templates from a simple monolingual corpus or from a parallel corpora, we derived these templates from different ontologies. The second difference concerns to the used method to discover the translations. We do not extract the translations from a corpus, but from different linguistic resources. Thus, the compositional method

---

\textsuperscript{16}\url{http://www.fao.org/aims/ag_download.htm}

\textsuperscript{17}\url{http://www.google.com/translate_t}

\textsuperscript{18}\url{http://babelfish.altavista.com/}

\textsuperscript{19}\url{http://ets.freetranslation.com}
first searches for translation candidates of a given composed label and then builds the translations for the candidates using lexical templates.

The lexical templates derived from different ontologies are used to control the order of translation. The main steps of the algorithm are:

1. The compound label is normalized, e.g., rewriting in lowercase, hyphens are removed, it is split into tokens, etc (see segmentation task in the figure 8).

2. A set of possible translations is obtained for each token of the compound label using the different translation paradigms (first component in our translation strategy). Note that no process of combination is executed among different translations obtained from each method. The method uses all possible combinations of translation obtained for each token.

3. Since translations between languages do not keep the same word order, the algorithm creates candidate translations in the target language using lexical templates\textsuperscript{20}. Each lexical template contains at least a pair of patterns, namely ‘source’ and ‘target’ patterns. A source pattern is a template to be compared with the tagged compound label\textsuperscript{21}, described in the source language, while the target pattern is used to generate the label in the target language. If no applicable template is found, the compound label is translated using the translation service directly.

4. All the candidate labels that fulfill the target pattern are returned as candidate translations of the compound label.

In our approach, we used a semi-automatic process to obtain the lexical templates. As we explained before, each lexical template is composed of source and target patterns. The ontology labels used to learn the source patterns were extracted from different domain ontologies expressed in English, German, or Spanish. Each label was tokenized, and tagged using the language independent part-of-speech tagger proposed in [18]. On the other hand, the labels used to learn the target patterns were extracted from the multilingual information associated to each ontological term or by means of a manual translation process. The same process used to annotate part of speech (POS) in the labels of the source patterns was used to annotate the labels of the target patterns. A sample list of lexical templates learned to translate composed labels among English, Spanish and German can be consulted in [5].

5.3. Label Post-Processing

This component show the translations to the user for review the quality of them. The quality of the translations is measured by means two factors adequacy and fluency. The first factor is used to determine the quantity in that the meaning of a correct translation is preserved. While that the fluency factor is used to determine how good the corresponding translation in the target language is.

The checking of the quality of a translation is the only task of the ontology localization activity in which the user interacts necessarily. In the next versions of our system we will try automatize this component.

\textsuperscript{20}The notion of lexical template proposed in this paper refers to text correlations found between a pair of languages.

\textsuperscript{21}We use TreeTagger in order to annotate the compound labels with part-of-speech and lemma information.
6. Experimental Evaluation

In this section we describe a set of experiments that were conducted with the objective of evaluating LabelTranslator system. In section 6.1 we describe the experiments used to evaluate some aspects related to the translation ranking techniques, where the task is to select the most appropriate translation for ontology labels. Section 6.2 deals with the study used to assess the usability of the LabelTranslator system for carrying out the ontology localization activity.

6.1. Performance Evaluation

To evaluate the quality of translation of the LabelTranslator system we conducted in March 2008 a preliminary experiment [5] involving ten PhD students. The main goal of the experiment was to evaluate the translation ranking techniques used by the system to select the most appropriate translation for each ontology label. This was done by comparing the translations provided by an expert (gold standard) with the translations provided by the ranking algorithm used in LabelTranslator. The ontology corpus used for the evaluation was selected from the set of KnowledgeWeb22 ontologies used to manage EU projects. The experimental results showed that our system suggested the correct translation 72% of the times. Also, the values of recall obtained suggested that a high percentage of correct translations were part of the final translations shown to the user. One of the main limitations was the low quality of the translations of compound labels. We implemented some improvements to the algorithm as i) a recursive function that attempts to match the bi/tri-tokens of a compound label with the lexical templates23 stored in the database, or ii) a method that learns new lexical templates from the translations supplied by the user.

To test the improvements implemented, a new experiment was performed in the “Artificial Intelligence (AI)” Master course at the Facultad de Informática (Universidad Politécnica de Madrid) with 17 Master students. We decided to use a questionnaire that allowed collecting the assessments of the students about the capacity of the translation algorithm in providing correct translations according to the context. As a general conclusion of the results obtained, we can mention that 33% of the students identified the level of correctness of the translations greater than 80%. The rest of students believed that the translations obtained had a level of correctness greater than 90%. Basically, the main strength provided by this experiment was the improvement in the quality of translation of compound labels. However, some weaknesses were detected: i) the misuse or omission of the definite article; ii) the incorrect translation of acronyms; and iii) the erroneous identification of the part of speech (POS) of single words or compound labels. The final aspect is very important, because we use POS tagging as a first mechanism of disambiguation to discard candidate translations. We are now working to solve these problems. More details about this experiment in [4].

6.2. Usability Evaluation

To assess the usability of the LabelTranslator system we conducted an experiment following the Software Usability Measurement Inventory (SUMI) method [14]. The SUMI questionnaire includes 50 items for which the user selects one of three responses (“agree”, “don’t know”, “disagree”). The questionnaire is designed to measure the affect, efficiency, learnability, helpfulness, and control of a software product [10]. The experiment involved 10 participants, most of whom were PhD students with a good command on ontology engineering. The experimenters met with all participants for 10 minutes to explain the purpose of the evaluation session and present the methodology of SUMI evaluation. Then, the participants had 20 minutes to translate an ontology using the guidelines and LabelTranslator, and 10 minutes to fill in the SUMI questionnaire for user-interaction satisfaction. During these two phases of the experiment users were not allowed to ask questions to the evaluators.

The most important findings of the experiment are related to the high level of learnability shown by LabelTranslator, especially in the case of a novice user. There was only one evidence about the need of making minor modifications in the LabelTranslator user interface to improve affect and efficiency with better navigation and informative functions. Additional details about user perception with respect to the goals of each SUMI dimension can be found in [4].

---

22http://knowledgeweb.semanticweb.org/
23The notion of lexical template refers to text correlations found between a pair of languages.
7. Conclusion

In this paper, we have presented LabelTranslator, our approach to semi-automatically localize ontologies among English, Spanish, and German. We first have explained the generic architecture of our approach to support an automated ontology localization in collaborative and distributed environments. In order to define the infrastructure requirements we took as base the localization requirements of international institutions as FAO and then we compared them with our own observations in the field.

We have included the description of technical details of the main components in the architecture. So, we have discussed the different aspects concerning to the implementation of the Ontology Repository component included in the Ontology Management module. A description of the capabilities of the implemented interfaces has also included. Concerning to the Localization Management module we have described their main functionalities, that are: synchronize the changes of the ontology and linguistic model and implement the actions described in the collaborative workflow. Hence, we propose two generic workflows specialized at different levels of abstraction (i.e., ontology element level and translation level). Our proposal includes the definition of the role of the workflow in the infrastructure required for the management of the multilingual content for localization, and describes its relationships with other modules and activities (i.e., synchronization, label translations, etc.).

We have finished the description of technical details of the main components in the architecture by commenting some implementations details about Ontology Translator module. We have described here the translation strategies used for localize simple and compound labels.

Although we have already evaluated, the technological aspects of the individual components of our approach [8], the next step is the complete evaluation of our collaborative approach within FAO and other collaborative and distributed scenarios.

As future work, we point out current needs that are not addressed in this work and that will have to be addressed in future versions of LabelTranslator system:

- Evaluation. It is necessary to design extensive evaluation mechanism of ontology localization systems. Beside evaluating systems, it is necessary to be able to help users in choosing the appropriate translation technique or to combine the most appropriate techniques for their tasks. We have tried in this work to identify some localization strategies, but a lot remains to be investigated, for example for the localization of instances and ontology term translations.

- Translation techniques. The machine translation field identifies a wealth of basic techniques which can be used to discover the translations of the ontology elements. In this work we have used only some of these techniques to discover the translations. However, further investigation is necessary in order to incorporate for example corpus-based and web-based techniques in the localization activity.

- Translation strategies. In this work we have incorporated two basic translation strategies for discovering the translations of simple and compound labels. One of the important issues to deal with is the proper combination and integration of various categories of translators. In particular, the integration of corpus-based (statistical) and ontology-based (semantic) techniques is of high interest.

Acknowledgements

This work was funded by the Neon Project sponsored under EC grant number (FP6-248458) and it is actually supported by the Monnet Project (FP6-248458) and the CICYT project TIN2007-68091-C02-02.

References


