Intelligent Systems
IEEE
www.computer.org/intelligent

Mining the Web to Create Specialized Glossaries

Paola Velardi, Roberto Navigli, and Pierluigi D'Amadio

Vol. 23, No. 5
September/October 2008

Mining the Web to Create Specialized Glossaries

Paola Velardi, Roberto Navigli, and Pierluigi D'Amadio,
University of Rome "La Sapienza"

TermExtractor and GlossExtractor, two Web-mining-based applications, support glossary building by exploiting the Web's evolving nature to allow continuous updating of an emerging community's vocabulary.

A first step in establishing a Web community's knowledge domain is to collect a glossary of domain-relevant terms that constitute the linguistic surface manifestation of domain concepts. A glossary is a list of domain terms and their textual definitions. For instance, Yahoo's financial glossary defines *security* as a "piece of paper that proves ownership of stocks, bonds, and other investments." Glossaries are helpful for integrating information, reducing semantic heterogeneity, and facilitating communication between information systems. However, although established domains largely have glossaries, new communities typically have trouble providing them. Even when they're accessible, they often provide emerging knowledge in an unsystematic, incomplete manner.

Manually constructing glossaries requires the cooperative effort of a team of domain experts and involves several steps, including identifying consistent domain terminology, producing textual definitions of terms, and harmonizing the results. This procedure is time-consuming and costly,[1] often requiring the support of a collaborative platform to facilitate shared decisions among experts. It's also an incremental process that must be continuously iterated as emerging terms come to use or as authoritative definitions for existing terms in the domain are produced. To help alleviate the issues of manual construction, you can partly automate the process. Glossary extraction systems should speed up the three steps of manually constructing glossaries by

- minimizing the time spent actually building the domain terminology and glossary,
- ensuring continuous maintenance and updates, and
- providing support for collaborative validation and refinement of the glossary extraction results.

Unfortunately, existing glossary extraction systems don't truly meet these objectives. In this article, we present a comprehensive, fully engineered set of natural language tools that assist an online community in the entire domain glossary extraction process. (For a discussion of other tools and methodologies, see the "Related Work in Glossary Extraction" sidebar.) We based our methodology on machine learning and Web-mining techniques and implemented it in two tools, called TermExtractor (http://lcl.uniroma1.it/termextractor) and GlossExtractor (http://lcl.uniroma1.it/glossextractor). The tools acquire a glossary's two basic components: terms and definitions. The former are harvested from domain text corpora, and the latter are extracted from different types of Web pages (such as online glossaries and Web documents).

The tools use text-processing techniques to

1541-1672/08/$25.00 © 2008 IEEE
Published by the IEEE Computer Society

Related Work in Glossary Extraction

In the past few years, the Web has been a valuable resource for extracting several kinds of information (such as terminologies, facts, and social-networking information) to aid text-mining methods. Nonetheless, automatic-glossary-extraction literature, especially from the Web, isn't very rich. Atsushi Fujii and Tetsuya Ishikawa presented a method that extracts fragments of Web pages using patterns typically used to describe terms.[1] They used an *n*-gram model to quantify how well formed a given text fragment was. Although they used a clustering approach to identify the most representative definitions, they didn't perform domain-based selection. Youngja Park, Roy Byrd, and Branimir Boguraev's method, implemented in the GlossEx system, extracts glossary items (that is, a terminology) on the basis of a pipeline architecture.[2] This method performs domain-oriented filtering that improves over the state of the art in terminology extraction; nonetheless, it provides no means for learning definitions for the extracted terms. Judith Klavans and Smaranda Muresan's text-mining method, the Definder system, extracts definitions embedded in online texts.[3] The system is based on pattern matching at the lexical level and guided by cue phrases such as "is called" and "is defined as." However, as in Fujii and Ishikawa's method, the Definder system applies patterns in an unconstrained manner, which implies low precision when applied to the Web, rather than a domain corpus.

A closely related problem to glossary extraction is open-domain definitional question answering (QA), which focuses on unconstrained "who is X?" and "what is X?" questions. Many QA approaches—especially those evaluated in the TREC (Text Retrieval Conference, http://trec.nist.gov) evaluation competitions—require the availability of training data, such as large collections of sentences tagged as "definitions" or "nondefinitions." Applying supervised approaches[4,5] to glossary extraction in emerging domains would be costly and time-consuming. Ion Androutsopoulos and Dimitrios Galanis propose a weakly supervised approach in which feature vectors associated with each candidate definition are augmented with automatically learned patterns.[6] Patterns are *n*-grams that surround a term for which a definition is sought. Horacio Saggion's method extracts definitions from a large corpus using lexical patterns coupled with a search for cue terms that recur in dictionary and encyclopedic definitions of the target terms.[7]

A recent article proposes using probabilistic lexico-semantic patterns (called *soft patterns*) for definitional QA.[8] These patterns cope better with the diversity of definition sentences than predefined patterns (or *hard patterns*) can. The authors evaluated the method on data sets made available through TREC. However, we must outline some important differences of glossary extraction with respect to the TREC "what is" task.

First, in TREC, the target is to mediate at best between precision and recall; when the objective is an emerging domain, recall is the most relevant performance figure. For certain novel concepts, there might be very few definitions (or just one). The target is to capture the majority of them because rejecting a bad definition takes just seconds, whereas manually creating a definition takes several minutes. (We consulted a professional lexicographer, Orin Hargraves, to estimate the manual effort of glossary development.)

Second, TREC questions span heterogeneous domains—that is, there's no domain relation between definitions, in contrast with glossaries.

Third, the TREC evaluation data set doesn't have true definitions but rather sentence "nuggets" that provide some relevant fact about a target. The following example shows a list of answers, classified as "vital" or just "okay," for the query, "What is Bollywood?" from TREC 2003.

Qid 2201: *What is Bollywood?*

2201 1	vital	is Bombay-based film industry
2201 2	okay	bollywood is indian version of Hollywood
2201 3	okay	name bollywood derived from Bombay
2201 4	vital	second largest movie industry outside of hollywood
2201 5	okay	the bollywood movies in UK's top ten this year
2201 6	okay	empty theaters because films shown on cable
2201 7	okay	Bollywood awards equivalent to Oscars
2201 8	vital	700 or more films produced by india with 200 or more from bollywood
2201 9	vital	Bolywood—an industry reeling from its worst year ever

This example shows that manually validated answers (even those classified as "vital") aren't always true definitions (for example, answer 8) and aren't professionally written (for example, answer 9). Wikipedia gives a true definition of Bollywood: "Bollywood is the informal term popularly used for the Mumbai-based Hindi-language film industry in India." Given the TREC task objectives, the use of soft matching criteria is certainly crucial, but in glossary definitions, style constraints should be followed closely to provide clarity and knowledge sharing, as well as to facilitate validation and agreement among specialists.

In summary, the state of the art on glossary and definition extraction only partially satisfies the objectives of the glossary extraction life cycle. In the main article, we present our glossary extraction system, which employs natural language techniques to exploit the Web's potential.

References

1. A. Fujii and T. Ishikawa, "Utilizing the World Wide Web as an Encyclopedia: Extracting Term Descriptions from Semi-Structured Texts," *Proc. 38th Ann. Meeting Assoc. for Computational Linguistics,* Morgan Kaufmann, 2000, pp. 488–495.
2. Y. Park, R. Byrd, and B. Boguraev, "Automatic Glossary Extraction: Beyond Terminology Identification," *Proc. 19th Int'l Conf. Computational Linguistics,* Howard Int'l House and Academica Sinica, 2002, pp. 1–7.
3. J. Klavans and S. Muresan, "Evaluation of the Definder System for Fully Automatic Glossary Construction," *Proc. American Medical Informatics Association Symp.,* 2001, pp. 324–328.
4. S. Miliaraki and I. Androutsopoulos, "Learning to Identify Single-Snippet Answers to Definition Questions," *Proc. 20th Int'l Conf. Computational Linguistics,* Morgan Kaufmann, 2004, pp. 1360–1366.
5. H.T. Ng, J.L.P. Kwan, and Y. Xia, "Question Answering Using a Large Text Database: A Machine Learning Approach," *Proc. Conf. Empirical Methods in Natural Language Processing,* Assoc. for Computational Linguistics, 2001, pp. 67–73.
6. I. Androutsopoulos and D. Galanis, "A Practically Unsupervised Learning Method to Identify Single-Snippet Answers to Definition Questions on the Web," *Proc. Human Language Technology Conf. and Conf. Empirical Methods in Natural Language Processing,* Assoc. for Computational Linguistics, 2005, pp. 323–330.
7. H. Saggion, "Identifying Definitions in Text Collections for Question Answering," *Proc. Language Resources and Evaluation Conf.*, European Language Resources Assoc., 2004.
8. H. Cui, M.K. Kan, and T.S. Chua, "Soft Pattern Matching Models for Definitional Question Answering," *ACM Trans. Information Systems,* vol. 25, no. 2, 2007, pp. 1–30.

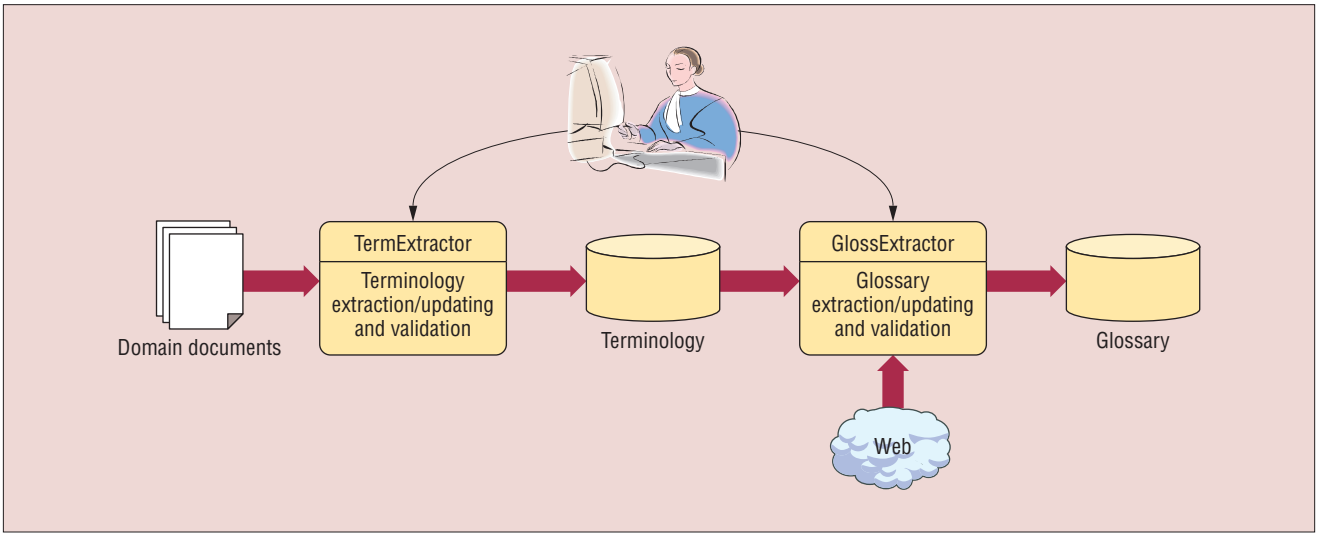


Figure 1. The glossary extraction process. The process comprises two phases—terminology extraction and glossary extraction.

analyze both the documents' content and structure. They exploit the Web's highly evolving nature to allow for continuous, incremental updating of the domain lexicon and the corresponding definitions. On one side, in emerging communities, terms often gain and lose popularity in a short time. On the other side, new definitions of established terms might become available over time.

System architecture

Figure 1 shows our glossary extraction process, which comprises two phases.

In *terminology extraction*, the user collects domain-relevant documents (such as a scientific community's publications, a commercial organization's product descriptions, technical manuals, or a company's best practices) and submits them to the TermExtractor system. This produces a domain terminology (that is, a flat list of terms) that the user receives for single-user or group validation. (We've described the terminology extraction phase in past work.[2])

In *glossary extraction*, the user submits terminology to the GlossExtractor system, which explores the Web for one or more definition sentences for each term. Finally, an evaluation session occurs, and the validated glossary is ready for download. (We've previously described a basic version of GlossExtractor,[3] but both the filtering techniques and the experiments we present here are novel.)

The user can repeat the extraction process many times, incrementally, to update and maintain the glossary. The two Web applications give the user several options to tune the run for the community's needs, but describing these options in detail is outside this article's scope (although the interested reader can access the Web applications and follow the instructions).

TermExtractor

To be complete, and to provide a clear picture of the complete glossary-learning procedure, we summarize the system's algorithms and features here.

First, the user sets a list of options (including minimum and maximum length of terminological strings, named-entity detection, updating of an existing terminology or creation of a new one, single-user or collaborative validation, and many others) or accepts the default settings.

Second, the user submits a set of documents in any common format (such as doc, txt, pdf, or latex) either one by one, or as a compressed archive.

Third, the terminology extraction process begins by applying part-of-speech tagging and chunking to the input documents (we used TreeTagger; www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger). Consequently, the tool selects nominal chunks (typically including adjectives and nouns) as term candidates. Filtering of domain-pertinent terminological strings from extracted candidates is based on the following measures:[2]

- *domain relevance*, a measure for identifying patterns that are frequent in the domain corpus and infrequent in a predefined user-adjustable set of contrastive domains;
- *domain consensus*, an entropy-based measure that determines whether terms are evenly distributed across the domain documents;
- *lexical cohesion*, a measure that aims to discriminate between the chance of co-occurrence and real semantic cohesion by analyzing the component words of candidate terms;
- *structural relevance*, a set of heuristics that assign a higher weight to terms occurring in specific formatting structures, such as titles, abstracts, or boldface; and
- *heuristics*, a set of additional heuristic rules that prune out certain strings, such as those including very common adjectives, numbers, or strings. Heuristic rules also extract single-word terms, which are often subject to noise.

Finally, when extraction and filtering are complete, the tool automatically sends an email to the user, directing him or her to the validation page, which supports both single-user and group validation.

As an example, Figure 2 shows the result of terminology extraction when submitting a draft version of this article. Terms can be marked for rejection (as done in the figure for the term "threshold embed"). When validation completes, the user can download the result in one of several formats (such as plain text, Excel, or XML).

TermExtractor has been online for nearly two years and has been evaluated both in the context of the Interop European project (www.interop-vlab.eu) and by many users

across the world. We performed an evaluation on different domains involving 12 different research and industrial institutions, which led to an average precision of 80 percent in terminology extraction.[2]

GlossExtractor

Here, we describe our glossary extraction methodology and its implementation as a Web application.

The gloss extraction algorithm

The algorithm takes as input a list T of terms and outputs a (possibly empty) set of definitions for each term $t \in T$. Possibly, T is the result of the previous terminology extraction process.

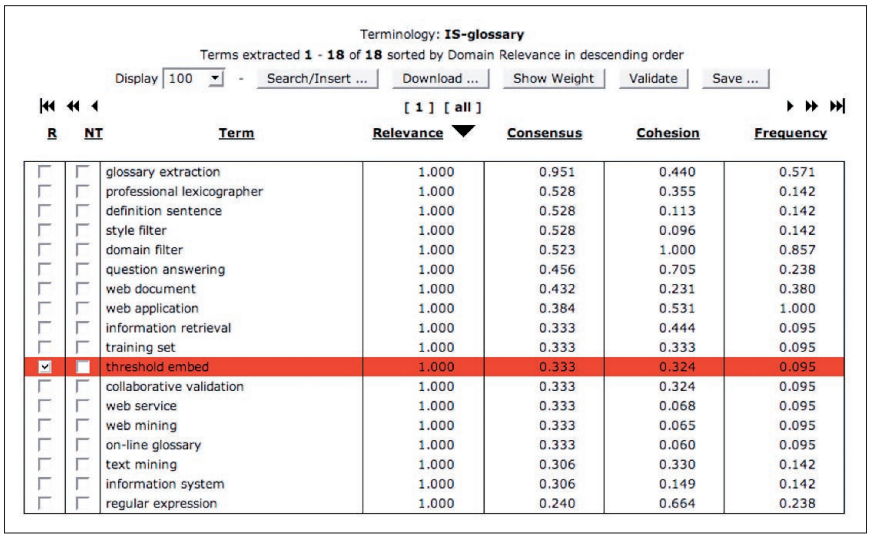
The algorithm comprises three phases: candidate extraction, stylistic-based filtering, and domain-based filtering.

Candidate extraction. First, for each term $t \in T$, the tool searches for definition sentences in online glossaries and Web documents. To retrieve candidate definitions in Web documents, it queries Google with the following set of manually defined patterns:

> "*t* is a," "*t* is an," "*t* are the," "*t* defines," "*t* refers to," "*t* concerns," "*t* is the," "*t* is any"

These patterns, inspired by previous work on lexico-syntactic patterns,[4] are intentionally very simple to reduce search and retrieval time. The tool saves the first five pages returned for each query (and converts them to plain text format). Pages are retrieved on the basis of a single text snippet matching one of our patterns. However, other potential definitions might be available in the document. Thus, from each page we extract all the sentences (that is, our candidate definitions) that contain our target term *t*. This strategy results in a considerable amount of noise in our corpus of candidate definitions. Noise typically is generated by two factors:

- *Nondefinitional sentences.* Given their generality, our simple patterns match definitional as well as nondefinitional sentences, such as, "Bollywood is a great inspiration for young actors." The "X is a" pattern is a major source of noise, but many definitional sentences follow this style. In other words, this pattern's precision is low, but its recall is high. To remove nondefinitional candidates matching our regular expressions, we resort to a stylistic filter.
- *Out-of-domain definitions.* The sentence could indeed be a definition, but one that isn't pertinent to our domain. For example, "A model is a person who poses for a photographer, painter, or sculptor" is pertinent to the fashion domain but not to the software engineering domain. To discard out-of-domain definitions, we use a domain filter.


Terminology: **IS-glossary**

Terms extracted **1** - **18** of **18** sorted by Domain Relevance in descending order

Display 100 - Search/Insert ... | Download ... | Show Weight | Validate | Save ...

[1] [all]

R	NT	Term	Relevance	Consensus	Cohesion	Frequency
☐	☐	glossary extraction	1.000	0.951	0.440	0.571
☐	☐	professional lexicographer	1.000	0.528	0.355	0.142
☐	☐	definition sentence	1.000	0.528	0.113	0.142
☐	☐	style filter	1.000	0.528	0.096	0.142
☐	☐	domain filter	1.000	0.523	1.000	0.857
☐	☐	question answering	1.000	0.456	0.705	0.238
☐	☐	web document	1.000	0.432	0.231	0.380
☐	☐	web application	1.000	0.384	0.531	1.000
☐	☐	information retrieval	1.000	0.333	0.444	0.095
☐	☐	training set	1.000	0.333	0.333	0.095
☑	☐	threshold embed	1.000	0.333	0.324	0.095
☐	☐	collaborative validation	1.000	0.333	0.324	0.095
☐	☐	web service	1.000	0.333	0.068	0.095
☐	☐	web mining	1.000	0.333	0.065	0.095
☐	☐	on-line glossary	1.000	0.333	0.060	0.095
☐	☐	text mining	1.000	0.306	0.330	0.142
☐	☐	information system	1.000	0.306	0.149	0.142
☐	☐	regular expression	1.000	0.240	0.664	0.238

Figure 2. Terminology extraction results from a draft version of this article.

The two subsequent steps of our glossary extraction methodology, which we describe next, will help eliminate these sources of noise.

Stylistic filter. The stylistic filter aims to select well-formed definitions—that is, definitions expressed in terms of *genus* (the *kind* a concept belongs to) and *differentia* (what specializes the concept with respect to its kind). For instance, in the definition, "Database integration is the process of integrating systems at the database level," *process* is the genus, and the rest of the sentence is the differentia. Not all definitions are well formed in this sense (for example, "component integration is obtained by composing the component's refinement structures together"), and many sentences that are ill-formed aren't even definitions (for example, "Component integration has been recently proposed to provide a solution for those issues").

Stylistic filtering is a novel criterion with respect to the related definition extraction literature, and it has several advantages:

- to prefer definitions adhering to a uniform style, commonly adopted by professional lexicographers;
- to distinguish between definitions and nondefinitions (especially when candidate definitions are extracted from free texts, rather than glossaries); and
- to extract from definitions a kind-of information, used to arrange terms taxonomically (for details on automatic taxonomy construction, see our previous work[3,5]).

To learn a stylistic filter for well-formed definitions, we employed a decision tree machine learning algorithm. We used the J48 algorithm from the weka (www.cs.waikato.ac.nz/ml/weka) suite of machine learning software, trained on a data set of positive and negative examples from four domains—arts, tourism, computer networks, and enterprise interoperability (our *training set*, or TS). We acquired positive examples from professional online sources, whereas we obtained negative examples by manually collecting nondefinitional sentences, including the domain terms from the four domains, from Web search results and available data of a previous experiment conducted with the Interop project members. Table 1 illustrates the training-set composition. To make it a suitable input to the decision tree algorithm, we tagged and chunked each sentence in the TS sentences by parts of speech and transformed them into a vector of five elements, one for each of the first five part-of-speech/chunk tag pairs. For instance, our earlier definition of

Table 1. Training set used to learn a stylistic filter for definition sentences.

Domains	Arts*	Tourism†	Computer networks‡	Enterprise interoperability§
Positive	310 + 415	270 + 270	450	1,435
Negative	80	60	50	2,032

*AAT (www.getty.edu/research/conducting_research/vocabularies/aat) and WordNet (http://wordnet.princeton.com)
†WordNet (http://wordnet.princeton.com) and STB (http://app.stb.com.sg/asp/tou/tou08.asp)
‡Geek.com (www.geek.com/glossary)
§Interop glossary (http://interop-vlab.eu/backoffice/IEKR/iekr.2007-05-31.1222018345/?searchterm=glossary)

database integration produced the following output:

> Database integration is the process of integrating systems at the database level
>
> [NN NN] VBZ [DT NN] IN [VBG NNS] IN [DT NN NN],

where we enclose a chunk in square brackets (for example, the first chunk includes the first two nouns). Next, we transform the definition into a vector of the first five elements (we exclude the term being defined), plus the instance classification ("yes" for positive, "no" for negative):

> ((VBZ, -), (DT, chunk2), (NN, chunk2), (IN, -), (VBG, chunk3), YES).

As a result of training, we learn a decision tree model that we can use to filter future candidate definitions according to their style. Although the recognition of stylistic patterns is triggered by a supervised-learning algorithm, note that

- style is mostly a domain-independent criterion, so the training phase doesn't need to be repeated; and
- we needed limited manual effort to create the training set.

Domain filter. The domain filter prunes out candidate definitions that aren't pertinent to the domain of interest. We obtain a probabilistic domain model by analyzing the domain terminology. As we explained earlier, the input to the glossary extraction algorithm is a terminology T—that is, a flat list of single-word and multiword terms. Starting from the set of single words constituting the terminology T, we learn a probabilistic model of the domain, which assigns a probability that each single word occurs in the terminology T. Let T be the set of domain terms, and let W be the set of single words appearing in T. Given a word $w \in W$, we estimate its probability of occurring in our domain of interest as

$$P(w) = \frac{freq(w)}{\sum_{w' \in W} freq(w')},$$

where $freq(w)$ is the frequency of w in terms of T. For example, if $T = \{$*distributed system integration, database integration*$\}$, then $W = \{$*distributed, system, integration, database*$\}$ and $P($*integration*$) = 2/5$ because out of five single words in T, *integration* appears twice.

Now, given a term t, let $Def(t)$ be the set of candidate definitions for t, and let $d \in Def(t)$ be a candidate definition for t. Finally, let W_d be the set of single words in d that also occur in W ($W_d \subseteq W$)—that is, W_d is the set of single words occurring both in d and in our initial terminology T. We define d's *domain pertinence* as a function of the occurrence probabilities of the terminological words occurring in d and those composing term t. Formally, we define the formula as

$$DP(d) = \sum_{w \in W_d} P(w) \log \frac{N_t}{N_t^w} + \alpha \sum_{w \in t} P(w)$$

where N_t is the number of definitions extracted for t (that is $N_t = |Def(t)|$), and N_t^w is the number of such definitions including the word w. The log factor, called inverse document frequency in information retrieval literature, gives less weight to words that have a very high probability of occurring in any definition, regardless of the domain (for example, "system"). The additional sum in this formula gives more weight to those sentences that include some of the words that compose the term t to be defined (α is an adjustable parameter).

For example, the following definition of *database integration* includes two occurrences of the term's words (marked in italics—we don't count the target term), plus the word *system* (with regard to the previous example of set W):

> Database integration is the process of *integrating systems* at the *database* level.

The system applies the measure of domain pertinence we defined earlier to candidates that the decision tree classifies as definitions. Finally, it applies a *term-dependent* threshold ϑ to each definition to select only those definitions d for which $weight(d) \geq \vartheta$. The system computes the threshold as the difference between the maximum and minimum values of domain pertinence of the definitions in $Def(t)$ averaged over the number of definitions.

Web application

We've implemented this methodology in the GlossExtractor Web application (accessible fromhttp://lcl.uniroma1.it/glossextractor). First, the user uploads a terminology T (possibly the result of a TermExtractor run) or runs a demo session, where he or she specifies only one term. If users upload a terminology, they can select a number of options, including the sources from which glossaries must be extracted (such as Web glossaries and Google's "define" feature and Web pages, which GlossExtractor searches). As with TermExtractor, the user can select single-user or group validation. Finally, the user assigns a name to the glossary and starts the extraction task. (Gloss search over the Web is time consuming because often several thousand queries must be issued. Furthermore, there's a limit to the daily number of queries; we thank Google's team for letting us extend this limit. Depending on the terminology's size and the number of active users, the process might not complete in one day, although usually it's a matter of dozens of minutes).

The glossary extraction process is asynchronous: the Web application ends and, as soon as the process is terminated, the user receives an email and is directed to a validation page. Figure 3 shows a screen dump of a validation page for the term "Bolly-

Table 2. The algorithm's performance (%) when searching Web glossaries and Web documents.

Data set	Number of glosses	Accuracy	Precision	Recall	F1 measure
Web glossaries	1,978	86.2	89.7	88.9	89.3
Web documents	359	85.1	92.5	81.0	86.4

wood." The validation page lists both the definitions above the filtering threshold ϑ and those below. For each definition, the system shows the computed weight and the source type from which the definition has been extracted (a Web glossary, Google's "define," or a Web document). In our experiments, we noticed that the distribution of definition sources varied significantly depending on the domain's novelty.

By clicking on the pencil icon to the left of each definition, the user can modify a definition's text if the definition is good but not fully satisfactory. The system tracks manual changes.

In a group validation, the coordinator has a different view of the validation page, where he or she can inspect the global votes each definition receives and make the final decisions.

#	R	Term	Edit	Gloss	Type	Confidence
1	☐	Bollywood	✏	Bollywood is an equivalent industry to Hollywood.	WEB_DOC	0.05000
2	☐	Bollywood	✏	Bollywood refers to Hindi films that are mostly made in Mumbai/Bombay.	WEB_DOC	0.05000
3	☐	Bollywood	✏	Successful Indian cinema industry - a pun on Hollywood.	GOOGLE_DEFINE	0.05000
4	☐	Bollywood	✏	Name of the Indian movie industry.	GOOGLE_DEFINE	0.05000
5	☐	Bollywood	✏	Bollywood refers to the name given to the largest film industry in the world, concentrated in Bombay.	WEB_DOC	0.05000
6	☐	Bollywood	✏	Bollywood refers to the popular cinematic style in India, which produces over 800 films annually.	WEB_DOC	0.05000
7	☐	Bollywood	✏	the film industry of India	GOOGLE_DEFINE	0.05000
8	☐	Bollywood	✏	Bollywood is the commonly used name for the the largest film industry in the world, that of India.	WEB_DOC	0.05000
9	☐	Bollywood	✏	Bollywood Is The Best.	WEB_DOC	0.00000
10	☐	Bollywood	✏	BOLLYWOOD IS a step ahead.	WEB_DOC	0.00000
11	☐	Bollywood	✏	Bollywood is an idea, perhaps a sensibility.	WEB_DOC	0.00000
12	☐	Bollywood	✏	Bollywood is de bijnaam van de Hindi filmindustrie in Bombay.	WEB_DOC	0.00000
13	☐	Bollywood	✏	Bollywood is not just something that appears on cinema screens and musical stage in this country, it is also a subject for Questico.	WEB_DOC	0.00000

Figure 3. Screen dump of a validation session. Highlighted definitions are those the system proposes to reject.

Evaluation

We used TermExtractor and GlossExtractor in the INTEROP European project,[3] and because they're freely available, several users around the world are using them in experiments in various domains. Here, we describe a set of experiments that have measured the glossary extraction methodology's performance. Few data are available on the evaluation of glossary extraction systems. Large-scale evaluations are available for the TREC (Test Retrieval Conference) task, but as we've explained, we have reasons for not using these data sets.

We conducted the evaluations as such: first, we performed a direct evaluation of GlossExtractor to assess our stylistic and domain filters' performance on a gold-standard data set. Then we conducted a real-world system evaluation, following the entire terminology and glossary extraction process from the Web.

Direct evaluation

To perform a direct objective evaluation of the core system's methodology, we assembled a set of definitions from online glossaries and Web documents. To obtain examples of good definitions, we extracted definitions from professional glossaries on the Web (partially from the same sources as for the learning set in Table 1, but choosing different definitions). Let G_1 be this initial set. Then, we used GlossExtractor to search the Web for additional definitions of the terms in G_1. We manually compared the set of extracted definitions, G_2, with G_1 to reliably separate good and bad definitions. Let G_{2g} and G_{2b} be the good and bad definitions, respectively. The resulting test set is then $TS = G_1 \cup G_{2g} \cup G_{2b}$.

Because good definitions (at least those in G_1) are professionally created, the test set is reasonably close to a gold standard. Using this test set, we ran several experiments to evaluate the system's ability to

- correctly classify good and bad definitions, when definitions are extracted only from glossaries;
- correctly classify good and bad definitions, when these are extracted only from Web documents; and
- prune out definitions that are good but not pertinent to the selected domain.

While the first two experiments evaluate the entire glossary extraction system (both style and domain filters are applied), the third experiment focuses on the domain filter's ability to discard out-of-domain definitions. Table 2 shows the result of the first two experiments. We assessed the system's accuracy, precision, recall, and F1 measure (the latter three calculated on good definitions). Both experiments highlight encouraging results, even in comparison with the few data available in the literature.[6]

To perform the third experiment, we created a test set of 1,000 economy definitions and 250 medicine definitions. We applied our domain filter and ordered the set of definitions, based only on their value of domain pertinence, computed on the economy terminology. This experiment was sufficiently realistic because, most of the time, technical terminology is meaningful in a restricted set of domains. The system generates an ordered list of definitions, where the first nonpertinent definition (for example, a medical definition) is found in position 571, and only 72 economy definitions appear in positions 571 to 1,000. So, regardless of the threshold value, the domain filter performs a good separation between pertinent and nonpertinent definitions. Of course, with an appropriate selection of the threshold, it's at best possible to balance precision and recall; however, in new domains, high recall is often preferable to high precision. Also, the domain

Table 3. The extraction algorithm's performance on a live search with postevaluation.

Statistics	Interoperability	Medicine
Total number of submitted terms (*T*)	100	100
Total number of extracted sentences (from all sources) (*E*)	774	1,137
Sentences over the threshold (*C*)	517	948
Sentences over the threshold accepted by evaluators (*A*)	448	802
Terms with at least one positive definition (*N*)	51	96

Performance measures	Interoperability	Medicine
Precision on positive (*A*/*C*)	86.85	84.59
Coverage (N/T)	51.00	96.00

Table 4. Glossary extraction performance for the terms in Figure 2.

	Positive	Negative	Total
True	14	106	120
False	5	25	30
Total	19	131	150

Precision on positive	73.68	(14/19)
Precision on negative	80.92	(106/131)
Accuracy	80.00	(120/150)
Coverage	64.70	(11/17)

filter's ability to discard inappropriate definitions depends merely on the domain model acquired exclusively on the basis of the input terminology *T*. Thus, no knowledge of other domains is requested to perform this task.

Real-world evaluation

Table 2 provides a reasonably objective evaluation of the system, given that "good" examples come mostly from professional glossaries or have been compared with professional definitions. However, we didn't perform the evaluation on a real task because we asked the system to analyze a predefined set of candidate definitions and to classify them as good or bad using the style and domain filters.

To obtain an evaluation more realistic to the system's intended use, we ran a real-world experiment in which the system actually searched the Web. In this case, we performed validation manually using a "three judges with adjudication" approach (that is, the adjudicator made the final choice in case of disagreement between the judges). To exploit our experience, we selected an enterprise interoperability domain. We repeated the experiment on a medical domain because we also had expertise in this domain.

In this experiment, we provided the system with only a set of terms, and all definitions were automatically extracted from online glossaries or Web documents. Table 3 shows the results. Overall, the system performed similarly to the gold standard we used earlier (see Table 2). However, in this case we couldn't measure the recall (which would require annotating all candidate definitions available online).

Performance is similar across the two domains, but coverage (that is, the percentage of terms for which at least one good definition was found) is considerably lower for interoperability terms, as expected. In new or relatively recent domains, it's more difficult to find definitions, both in glossaries and in Web documents.

Finally, to test the complete terminology- and glossary-learning procedure, we performed a small experiment that any interested reader might replicate. First, we submitted a draft (extended) version of this article to TermExtractor. (When you submit a single document to TermExtractor in demo mode, it is automatically partitioned into sections [depending on the document length] to simulate a corpus and compute a consensus measure.) Figure 2 shows the results: TermExtractor extracted 18 terms, of which only one was rejected (probably owing to a file conversion error). Also, a few of the terms weren't so relevant to this article's research domain (for example, "professional lexicographer"). However, TermExtractor best applies its domain consensus filter to more than one document at a time.

After validation, TermExtractor submitted the terminology file of 17 terms to Gloss Extractor. The system produced 150 definitions (65 from Web glossaries and 85 from Web documents). Table 4 summarizes the evaluation's results. The system found definitions for all but four of the 17 terms: professional lexicographer, definition sentence, style filter, and domain filter. However, owing to false positives, some additional terms had poor definitions (although we found good definitions among those rejected, which we counted as false negatives).

The terms "style filter" and "domain filter" certainly are relevant to the domain, but because they're new terms that we defined, GlossExtractor couldn't find a definition for them on the Web.

To the best of our knowledge, no other system in the literature covers the

entire glossary extraction process (acquisition of a domain terminology, extraction of definitions, evaluation, updating, and maintenance). Our methodology exploits the evolving nature of the Web, which is seen as a huge corpus of texts that is processed to create and continuously update specialized glossaries. In fact, the procedure can be repeated incrementally with the same average effort-per-term, both to extend the set of definitions for existing terms as well as to extract definitions for new emerging terms. Therefore, the complete procedure is under the control of the interested community, including the glossary maintenance and updating problem. According to our experiments, especially those in the context of the INTEROP European project, the average speedup factor with respect to manual glossary building varies from one-fifth to one-half of the time employed by a professional lexicographer.

Real-world evaluation of GlossExtractor showed that coverage might vary significantly, depending on the domain's novelty. We've identified three strategies that could increase coverage: including other formats, such as non-HTML documents, in our Web searches; letting users explicitly provide URLs of existing online glossaries or document archives for the system to search; and extending our style filter using a much larger training set composed of Wikipedia definitions. We will include these features in a future release of GlossExtractor.

More important, we're studying the development of novel, motivated algorithms that we hope will further improve our style and domain filters. A comparison of their performance with our current implementation will eventually reveal whether textual definitions' style might vary (slightly) depending on the domain at hand.

Acknowledgments

We thank Orin Hargraves for his very helpful professional advice.

References

1. E.P. Bontas and M. Mochol, "Towards a Cost Estimation Model for Ontologies," *Proc. 3rd Berliner XML Tage,* Humboldt-Universität zu Berlin and Freie Univ. Berlin, 2005, pp. 153–160.
2. F. Sclano and P. Velardi, "TermExtractor: A Web Application to Learn the Common Terminology of Interest Groups and Research Communities," *Proc. 9th Conf. Terminology and Artificial Intelligence,* 2007.
3. P. Velardi, R. Navigli, and M. Pétit, "Semantic Indexing of a Competence Map to Support Scientific Collaboration in a Research Community," *Proc. 20th Int'l Joint Conf. Artificial Intelligence,* AAAI Press, 2007, pp. 2897–2902.
4. M.A. Hearst, "Automatic Acquisition of Hyponyms from Large Text Corpora," *Proc. 14th Int'l Conf. Computational Linguistics,* Assoc. for Computational Linguistics, 1992, pp. 539–545.
5. R. Navigli and P. Velardi, "Learning Domain Ontologies from Document Warehouses and Dedicated Web Sites," *Computational Linguistics,* vol. 30, no. 2, 2004, pp. 151–179.
6. J. Klavans and S. Muresan, "Evaluation of the DEFINDER System for Fully Automatic Glossary Construction," *Proc. Am. Medical Informatics Assoc. Symp.,* 2001, pp. 324–328.

The Authors

Paola Velardi is a full professor in the Department of Computer Science at the University of Rome "La Sapienza." Her research interests include natural language processing, machine learning, and the Semantic Web. Velardi received her Laurea degree in electrical engineering from "La Sapienza." Contact her at velardi@di.uniroma1.it.

Roberto Navigli is an assistant professor in the Department of Computer Science at the University of Rome "La Sapienza." His research interests include natural language processing and the Semantic Web and its applications. Navigli received his PhD in computer science from "La Sapienza." Contact him at navigli@di.uniroma1.it.

Pierluigi D'Amadio is an IT consultant in system architecture, software design, and telecommunication systems. His research interests are intelligent systems, natural language processing, and information retrieval techniques. D'Amadio received his degree in computer science from the University of Rome "La Sapienza." Contact him at damadio@di.uniroma1.it.