Integrated Document Browsing and Data Acquisition for Building Large Ontologies

Felix Weigel, Klaus U. Schulz, Levin Brunner, and Eduardo Torres-Schumann

Centre for Information and Language Processing (CIS)
University of Munich (LMU), Germany
{weigel,schulz,brunner,torres}@cis.uni-muenchen.de

Abstract. Named entities (e.g., "Kofi Annan", "Coca-Cola", "Second World War") are ubiquitous in web pages and other types of document and often provide a simplified picture of the document's content. We present an ontology currently containing 31,000 named entities in different languages from various domains such as history, geography, politics, sports, arts, etc., which is being developed at the University of Munich (LMU). The underlying graph data model is simple and yet extremely versatile in different application scenarios. We demonstrate a prototype of a graphical interface to both the ontology and to documents on the web or in a local document repository, with a tight interaction in both directions. Occurrences of concepts from the ontology are highlighted and hyperlinked in the documents. Unrecognized entities could be added to the database and related to other concepts in a semiautomatic process. The entity database can also be used for extending full-text queries on the web or the repository to semantically close documents, and for indexing different kinds of named entities in the document repository. Similar to a programming IDE, the system illustrates how integrated browsing, search and update functionality contributes to the construction of high-quality ontologies, fundamental to the vision of a truly "semantic" web.

1 Introduction

The Semantic Web is meant to enable reasoning not only on the contents of static web pages, but further and foremost on the underlying data sources such as databases, web services, document repositories, digital libraries, on-line encyclopaedias, etc. (sometimes called the "deep web"). A fundamental building block for the description, integration/mediation and inference on these heterogeneous data sources are *ontologies*, i.e., formal schemata of the concepts and relationships in one or more, possibly overlapping domains. Faced with a growing number of different knowledge representation formalisms [1,2] and tools for creating ontologies [3,4], we claim that in order to make *large* amounts of *existing* data accessible to the Semantic Web, formalisms and tools are needed which (1) make ontology development, maintenance and usage easy even for non-experts in knowledge representation, (2) strive for a feasible amount of inference rather than too much of fancy (and expensive) "magic", and (3) offer a data-driven approach for integrating ontologies and existing data sources.

We would like to comment on these controversial issues. First, experts are most apt at giving some thematic domain a precise structure, but non-expert users may not be able to query this complex structure, falling back to less sophisticated queries or simply browsing. By contrast, collaborative projects such as Wikipedia [5] clearly show the benefits of both expert and non-expert contributions for gathering encyclopaedic knowledge. Second, even in the presence of emerging knowledge representation standards such as OWL [1], it seems natural to study simple formalisms by experimenting with prototypes of a limited functionality, before coming up with more sophisticated applications in a second step. Third, we believe that a fine-grained, up-to-date model of common-sense knowledge on a web scale is infeasible. Fully automated knowledge extraction may complement manual and semiautomatic acquisition techniques, but cannot replace them. We thus advocate a mainly data-driven process where ontologies are updated and enlarged while searching and browsing the actual contents.

Contributions. In this work we describe the *EFGT Net*, an ontology containing currently 31,000 named entities from various domains, which is being developed at the University of Munich. The underlying graph data model is deliberately simple and specifically designed for building scalable models of common-purpose knowledge. The ontology features multilingual concept descriptions, which we consider indispensable in a web context. Target applications include simple semantic search and inference on web contents or documents from a local repository or intranet. We also outline a new integrated tool – similar to an Integrated Development Environment (IDE) – for searching, exploring and updating the ontology while browsing the documents. An extensive sample session illustrates typical usage patterns with a prototype, emphasizing the benefit of a tight ontology/corpus integration for creating and using large-scale, high-quality ontologies.

The next section describes in a nutshell knowledge representation and inference with the EFGT Net. Section 3 presents our prototype using a real-world web example. The system architecture is sketched in Section 4. Finally, Section 5 concludes with a brief outlook on future work.

2 Knowledge representation and inference with EFGT

This section introduces the EFGT Net and its basic principles on an informal basis. For a more formal definition of the data model, see [6]. The EFGT Net is a directed acyclic graph (DAG) whose nodes represent concepts and whose edges represent directed binary relations between the concepts. Each concept has a natural-language definition in at least one of the supported languages, as well as optional semantic and syntactic information. In the current version of the EFGT Net there are about 31,000 nodes and 637,000 edges, with a language coverage of 100% German, 70% English, 30% French, 30% Polish and 10% Bulgarian. Every node has a unique identificator, its *ID string*, which determines the position of the corresponding concept in the structure of the EFGT Net and all its relations to other concepts. Thus from the complete set of identificators, all the edges can be inferred by means of a couple of formal deduction mechanisms [7].

The EFGT Net is built and enlarged by generating ID strings according to a set of formal rules. Syntactically, ID strings are defined by the grammar in Fig. 1. The seven alternatives in the second production can be arranged into the four main types E, F, G and T (from which the acronym EFGT is derived). Uppercase letters denote sets and lowercase letters singleton elements:

$\mathbb{I}$:= `(` `)` | `(` $\mathbb{X}\mathbb{I}$ `.` $\mathbb{N}$ `)` | `(` $\mathbb{I}$ `&` $\mathbb{I}$ `)`
$\mathbb{X}$:= `(` `e` | `E` | `F` | `g` | `G` | `t` | `T` `)`
$\mathbb{N}$:= $\mathbb{D}$ `(` `0` | $\mathbb{D}$ `)`*
$\mathbb{D}$:= `(` `1` | ... | `9` `)`

Fig. 1. ID string syntax.

- E, e Type E denotes a set of Entities like *composers*, whereas type e denotes a singleton entity like *J. S. Bach*.
- F Type F denotes a thematic Field (topic) like *politics*. Since every thematic field can be regarded as a set of subfields, there is no type f.
- G, g Type G denotes Geographical sets like *rivers*, whereas type g stands for singleton geographic sites like *the Alps*.
- T, t Finally, type T denotes a Temporal period like *epochs in art*, whereas type t denotes an individual time interval or point like *September 11th*.

As shown by the first production in Fig. 1, there are two ways to create a new ID string based on existing ID strings like the initial *top node* `()`. First one can refine a single existing ID string by a *local introduction* with the dot notation shown in Fig. 1. For instance, `(F().1)` defines the first concept below the top node, and is assigned to the topic "politics" in our ontology. When creating a new concept by local introduction, the only directly connected other concept is the existing concept being refined. The resulting edge can represent different relations, e.g., the definition of a subconcept – "foreign policy" `(F(F().1).2)` –, or subset – "political problems" `(F(F().1).20)` –, or the selection of a single member of a set ("Presidents of the United States"—"Bill Clinton"). This vagueness is intended to facilitate the modelling of real-world knowledge, mirroring the ambiguity of recursive constructs in natural language ("members of the Beatles" vs. "albums of the Beatles"). The second way to create a new ID string is to combine two existing ID strings with the `&` operator, which can either denote the intersection between two given sets, `(X&Y)`, or the creation of a new set `(X&y)` by placing the original set `X` in the context of a singleton concept `y`. Note that `(X&y)` need not be a subset of `X`. As an example combining local introduction and intersection, consider the concept "politics" `(F().1)` and its subfield "defense politics" `(F(F().1).14)`. The intersection with "persons" `(E().1)` yields the concept "defense politicians" `((F(F().1).14)&(E().1))`. For convenience, we henceforth drop occurrences of the top node, writing `(F.1)` for `(F().1)`, etc.

Inference with the EFGT Net

Figure 3 illustrates how the new concept "Cities in Bavaria" is represented in the ontology (for simplicity, ID strings are symbolized by their descriptions). Assume there exist concepts "Germany", "Bavaria" (a region in Germany), both of type g, and concepts "Cities", "Cities in Germany" of type E. "Bavaria" is derived from "Germany" by local introduction, whereas "Cities in Germany"

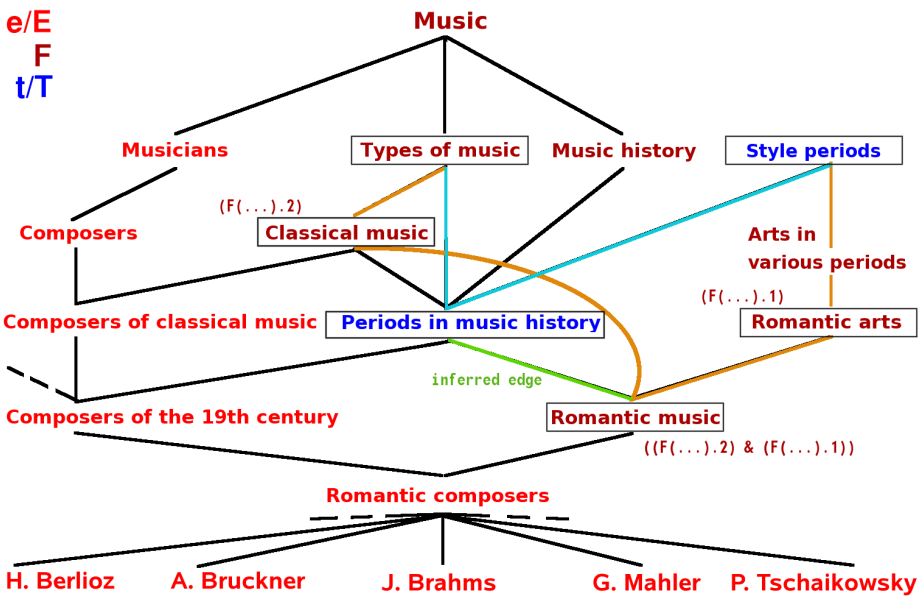

Fig. 2. Neighbourhood of the concept "Romantic composers" in the EFGT Net.

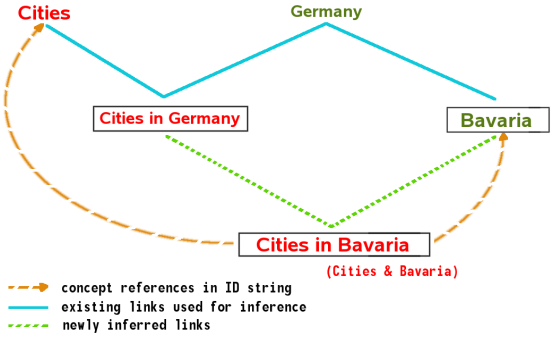

Fig. 3. Link inference in the EFGT Net.

results from combining "Cities" and "Germany" with the & operator. Analogously, the new concept "Cities in Bavaria" is defined simply by applying & to "Cities" and "Bavaria", as indicated by the dashed arrows. However, the edges actually inferred (green solid lines) relate the new concept directly to "Cities in Germany" (since Bavaria is a part of Germany), which implies an indirect link to "Cities". Typically many relations are inferred from a small number of references to existing concepts explicitly stated in the ID string. Figure 3 shows a similar case in a more complex subgraph. When inserting "Romantic music" into the graph, only "Classical music" and "Romantic arts" are explicitly mentioned in the ID string. The link stating that romantic music is a period in music history is inferred, based on the fact that the latter concept shares all higher-level ancestors ("types of music", "style periods") with the newly defined node. The exact inference algorithm is given in [7].

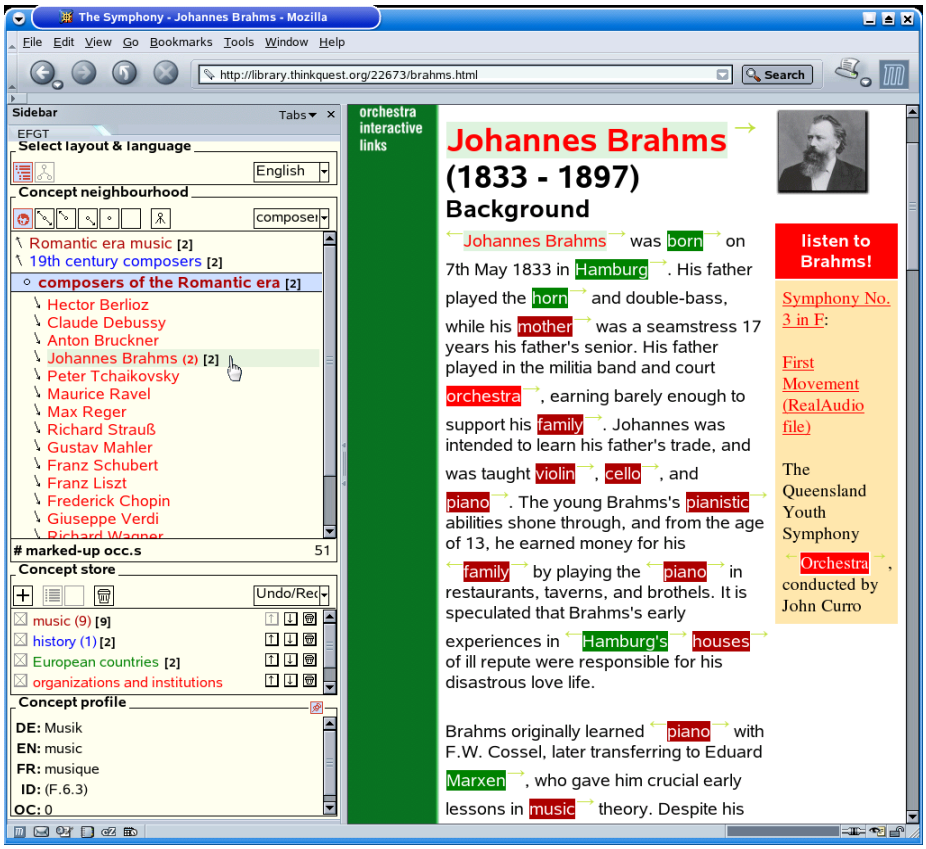

Fig. 4. Screenshot of the EFGT Net browser prototype.

3 Using the EFGT Net for Semantic Web browsing

This section describes a typical session with our prototypical ontology browser for the EFGT Net (see Figure 4). The system is explained in detail in Section 4. To achieve tight interaction of the documents and the ontology as claimed above, we integrated a GUI to the EFGT Net (left-hand side in Figure 4) into an ordinary web browser accessing documents in local repositories, intranets or the WWW (right-hand side). The EFGT GUI displays any concept along with its *neighbourhood*, i.e., the more general (specific) concepts one level above (below). In Figure 4, the concept "romantic composers" from Figure 2 is selected. Its ancestor (descendants) are labelled ↖ (↘). Colours indicate the EFGT type of all concepts as in Figure 2. Clicking on any item in the list shifts the focus to the neighbourhood of that concept. History and bookmark functions (above/below the list) and a backlink to the top node (🚶) facilitate browsing the EFGT Net.

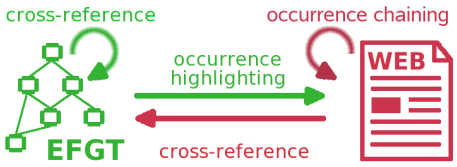

Fig. 5. Ontology interaction.

As outlined in Figure 5, the EFGT GUI interacts with the web page by highlighting selected occurrences of concepts in the document. The six highlighting modes (see the buttons on top of the neighbourhood) are: all EFGT concepts (), both more general and more specific (), more general/specific only (/), current concept only (), or no highlighting at all (). In Figure 4, the mode is active. A click on the button removes the highlighting of all occurrences visible in the screenshot, except "Johannes Brahms" and "music". Counters behind list items indicate how many occurrences of a concept itself (parentheses) or any of its descendants (square brackets) are highlighted in the document. The inference on ID strings for highlighting and counting ancestors or descendants is done on the fly with simple and fast string matching [6]. Occurrences of the same concept in the web page are chained through hyperlinks (light green arrows) for easy location of all paragraphs where a concept of interest is mentioned. Each occurrence is also backed by a hyperlink into the EFGT GUI, with the same functionality as links inside the GUI (focus shift, see above). Besides, by hovering an occurrence, the user obtains a *concept profile* (bottom left in Fig. 4) including, e.g., the life time of a person or the English description of the concept, which is useful when browsing documents in foreign languages.

Currently concepts in the GUI can be reached alternatively by (1) browsing the hierarchy down from the top node, (2) selecting a concept from the *concept store* containing bookmarks stored persistently in previous sessions, (3) selecting a concept visited earlier from the persistent history, or (4) clicking on an occurrence in the document. A simple string search on the natural-language concept definitions is being integrated into the prototype. Further extensions of the functionality, including ontology maintenance, are listed in Section 5.

4 Architecture of the prototype

The ontology browser presented in Section 3 is implemented as a client-server application using Java Server Pages (JSP) and Java Servlets, as sketched in Figure 6. The EFGT GUI is a JSP displayed in the sidebar of the web browser (Mozilla 1.7.12). Each focus shift (i.e., request for the neighbourhood of a specific concept) triggers the dynamic generation of a new web page containing the entire GUI shown in Figure 4. This includes (partly hidden) concept and widget labels in all supported languages, such that no reload is necessary when the user selects a different language from the drop-down list on top. The concept information needed to build the neighbourhood is obtained from the EFGT Net, which resides in a RDBS backend (PostgreSQL 8.0). Requests for a specific neighbourhood are answered through a Web Service interface from tables containing all nodes and edges in the graph. The underlying inference relies on simple and efficient string matching of ID strings. In order to reduce the response time even further, a server-side concept cache retains the most recently requested neighbourhoods.

The preparation of EFGT-enriched documents is a little more involved, as shown on the right-hand side of Fig. 6. The web browser directs each request for a new page (through a link or the address bar) to the servlet engine (Tomcat 5.0) running on a proxy server. The servlet receiving the request first fetches all data from the document repository or the web. Before the data is sent back to the client, all occurrences of EFGT concepts in the requested document must be located and hyperlinked using the ID strings from the EFGT Net, which are needed for the interaction with the EFGT GUI. We found that when implemented naïvely, preprocessing the documents thus on the fly easily entails inacceptable response times. Therefore we compile the required EFGT data into a set of rewrite lexica to be used by an extremely efficient and scalable string transducer [8]. Each lexicon contains the ID strings of all nodes in the EFGT Net, along with the concept definitions in a specific language. Thus the size of the lexica varies with the coverage for the corresponding languages. For instance, the German lexicon comprising all 31,000 nodes in our EFGT Net takes up 15 MB on disk. Note that the lexica are created off-line once (in a few seconds) and then used repeatedly when requests come in. Annotating all concept occurrences in a given document with EFGT data (and HTML markup for links) is a matter of milliseconds.

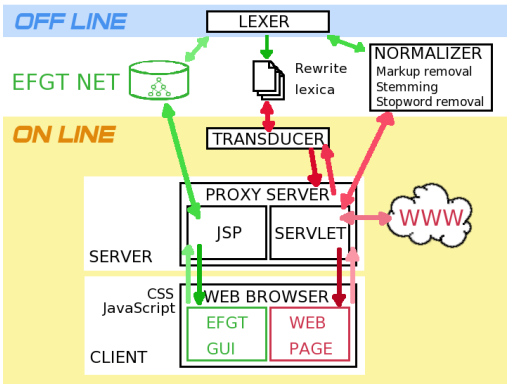

Fig. 6. Architecture of the prototype.

There are a couple of caveats related to the document preprocessing. First, to determine which lexicon to use for a given web page we examine the HTTP header *Content-Language*, using a simple fall-back heuristic based on the top-level domain of the requested URL if the header is missing. More sophisticated techniques could infer the language from the page contents. Second, markup must be temporarily stripped off the document before the transduction, otherwise HTML or embedded script code might get corrupted. Finally, to recognize inflected occurrences of EFGT concepts and to avoid needless annotation, some linguistic normalization (stemming, stop-word removal) is in order—both on-line in the document and off-line when creating the lexica. While this works fine for English content (using Porter stemming), a simple adaptation of the Porter algorithm to German resulted in many needless ambiguities. Hence we automatically generated all noun inflections of the German concept descriptions, added them to the lexicon (included in 15 MB), and disabled stemming for German.

5 Conclusions

In this paper we presented an integrated tool for document browsing with ontology support, and showed how the simple and versatile EFGT formalism provides efficient string-based inference on multilingual corpora such as local document

repositories or the web. The underlying data model addresses common-sense knowledge and can be universally applied and manipulated by non-experts. Efficient reasoning is achieved by comparing language expressions directly, which ensures the scalability of our approach. While the prototype presented here exploits inference only for simple subsumption tests, we envisage a more sophisticated semantic search and classification of the documents, using ID string manipulation either directly or with graphic metaphors. In the EFGT Net, semantic querying can be realized in an intuitive way without imposing to the user the burden of a complex formal query language. Further query refinement is achieved through interleaved ontology navigation and document browsing as illustrated above. The browsable GUI to the multilingual ontology distinguishes our system from other tools for automatic document annotation and hyperlinking, such as Magpie [9] and COHSE [10]. We plan to extend the prototype to support the user in ontology maintenance: e.g., new concepts occurring in the documents could easily be added on the fly to the EFGT Net using a drag-and-drop interface. By virtue of its simple recursive structure, the EFGT formalism seems particularly promising for such combined knowledge management and acquisition techniques. As the ontology grows, the automatic linking of entities and documents becomes more difficult since ambiguities arise. Apart from user-driven disambiguation (showing alternatives in context menus), we intend to develop detection methods for new entities and linguistic variants (maybe using named-entity recognizers such as GATE [11]), as well as intelligent indexing with on-line disambiguation. These steps should allow us to minimize human effort during ontology development.

References

1. Dean, M., Schreiber, G.: OWL Web Ontology Language Ref. (2005) W3C Rec.
2. Klyne, G., Carroll, J.J.: Resource Description Framework (2005) W3C Rec.
3. Sure, Y., Erdmann, M. et al.: OntoEdit: Collaborative Ontology Engineering for the Semantic Web. In: Proc. 1st Int. Semantic Web Conf. (2002) 221–235
4. Noy, N.F., Sintek, M. et al.: Creating Semantic Web Contents with Protege-2000. IEEE Intelligent Systems **16** (2001) 60–71
5. Wikipedia: The Free Encyclopedia. (`www.wikipedia.org`)
6. Schulz, K.U., Weigel, F.: Systematics and Architecture for a Resource Representing Knowledge about Named Entities. In: Proc. Workshop on Principles and Practice of Semantic Web Reasoning. (2003) 189–207
7. Brunner, L., Schulz, K.U., Weigel, F.: Organizing Thematic, Geographic and Temporal Knowledge in a Well-founded Navigation Space: Logical and Algorithmic Foundations for EFGT Nets. J. Web Serv. Research, Spec. Issue "Semantically Augmented Metadata for Services, Grids, and Software Engin." (2006) (in press).
8. Mihov, S., Schulz, K.U.: Efficient Dictionary-Based Text Rewriting using Subsequential Transducers. Journal of Natural Language Engineering (2005)
9. Dzbor, M., Domingue, J., Motta, E.: Magpie: Towards a Semantic Web Browser. In: Proc. 2nd Int. Semantic Web Conf. (2003) 690–705
10. Carr, L., Hall, W., Bechhofer, S., Goble, C.: Conceptual Linking: Ontology-based Open Hypermedia. In: Proc. 10th Int. World Wide Web Conf. (2001) 334–342
11. Cunningham, H., Humphreys, K. et al.: GATE – a General Architecture for Text Engineering. In: Proc. 5th Applied Natural Lang. Processing Conf. (1997) 29–30