Bootstrapping an Ontology-based Information Extraction System

Alexander Maedche[2], Günter Neumann[1], Steffen Staab[3]

[1]DFKI German Research Center for Artificial Intelligence,
Saarbruecken, Germany
neumann@dfki.de, http://www.dfki.de

[2]FZI Research Center at the University of Karlsruhe,
Karlsruhe, D-76131 Karlsruhe, Germany
maedche@fzi.de, http://www.fzi.de/wim

[3]AIFB, Univ. Karlsruhe, D-76128 Karlsruhe, Germany
staab@aifb.uni-karlsruhe.de,
http://www.aifb.uni-karlsruhe.de/WBS

Abstract. Automatic intelligent web exploration will benefit from shallow information extraction techniques if the latter can be brought to work within many different domains. The major bottleneck for this, however, lies in the so far difficult and expensive modeling of lexical knowledge, extraction rules, and an ontology that together define the information extraction system. In this paper we present a bootstrapping approach that allows for the fast creation of an ontology-based information extracting system relying on several basic components, *viz.* a core information extraction system, an ontology engineering environment and an inference engine. We make extensive use of machine learning techniques to support the semi-automatic, incremental bootstrapping of the domain-specific target information extraction system.

Keywords. Ontologies, information extraction, machine learning

1 Introduction

In order to overcome the problem of finding or extracting relevant information out of the enormous amount of text data electronically available, various technologies for information management systems have been explored within the Natural Language Processing (NLP) and AI community. One line of such research is the investigation and development of *intelligent information extraction* systems.

Information extraction (IE) is the task of identifying, collecting and normalizing relevant information from NL text and *skipping* irrelevant text passages. IE systems do not attempt an exhaustive deep NL analysis of all aspects of a text. Rather, they are built in order to analyse or "understand" only those text passages that contain information relevant for the task at hand. Thus, the IE system may be sufficiently fast and robust when dealing with free texts, such as appear on the Web.

The definition of relevancy is given implicitly by the IE model that specifies domain-specific lexical knowledge, extraction rules, and an ontology. The IE model allows to perform the required mappings from NL utterances to corresponding domain knowledge. In order to render possible an exhaustive and highly accurate extraction task, the model must be very detailed and comprehensive. Typically, the resulting mappings turn free text into target knowledge structures about crucial information — answering questions about *who*, *what*,*whom*, *when*, *where* or *why*.

The target knowledge structures are not arbitrary, but rather they are predefined by a given ontology. The ontology is a formal specification of a shared understanding of the domain of interest. For instance, in the field of market research, the ontology may describe concepts like revenue or joint venture, relations between concepts (e.g., company *has* joint venture), and axioms (e.g., the *owns* relationship between companies is transitive). Our current system is about tourism. Correspondingly, its ontology includes specifications of concepts like locations, accomodation, etc.
The overall functionality of the IE system may therefore be summarized by:

- Input:
  - IE model (Specification of lexical knowledge, extraction rules, and ontology)
  - A set of NL texts (e.g., press releases, online-documents, technical reports, or even email)
- Output: Target knowledge structure, i.e. a set of instantiated and related concepts and attributes. For instance, the IE system may assign to a web page about a hotel the structure (in pseudo code):
  h47 inst-of HOTEL, h47 HASNAME 'Schwarzer Adler',
  c28 inst-of CITY, c28 HASNAME 'Rostock', h47 LOCATEDIN c28.

This principal structure of IE systems offers itself to a wide range of important applications: intelligent information retrieval, text mining, e-mail routing, fine-grained text classification, automatic metadata generation, etc.

The major bottleneck of current IE technology lies in the adequate definition of lexical knowledge, extraction rules, and ontology such that a good coverage and precision is achieved by the extraction mechanism. The currently preferred approach is based on careful observations of relevant text corpora. In earlier systems (like the Fastus system, e.g., [2]), text corpora were analysed and the IE model has been defined by hand. The manual definition, however, turned out to be a very time consuming task. Thus, a number of machine learning approaches have been developed recently. They acquire parts of the IE model automatically from (various types of) text corpora (e.g., specifically annotated corpora or unannotated ones; cf. e.g., [12, 19, 22, 4], [8]). However, what is missing so far is an integrated approach for acquiring the IE model using machine learning techniques.

The ultimate objective that we pursue in our research is a fully integrated system for building an IE model and exploiting it in an IE system by applying a bootstrapping mechanism. Thereby, we found that the different parts of the domain model are reusable to different degrees. The domain-specificity increases when working from the specification of lexical knowledge, the specification of extraction rules and towards the definition of ontologies. Accordingly, the reusability decreases along the same line and the need for mechanisms that facilitate the specification starting from corpora grows.

Thus, there are quite reasonable resources now for lexical coverage and — to a lesser degree — for extraction rules, but virtually no ontologies that may be taken of the shelf. Therefore, we have approached the problem of acquiring the IE model on the ontology part.

We have built a comprehensive system that semi-automatically extends the ontology by ontology learning. It semi-automatically adapts and extends some of the lexical resources. It eventually maps free text, like web pages, onto ontology-based target knowledge structures. Its architecture was devised in order to allow for the construction and adaptation of even more resources, such as particular types of extraction rules, in the future.

The organization of the paper is as follows. We introduce the core components for bootstrapping an information extraction system. We start by introducing SMES (the Saarbücker Message Extraction System) and its core functionality and resources, i.e. lexical knowledge and extraction rules. Subsequently, we introduce the ontology components, namely the ontology engineering environment OntoEdit and the inference engine. The interaction between syntactic and conceptual level is then explained in Section 4. Finally, we outline the bootstrapping cycle exploiting the alternating use of the IE system and machine learning.

2 SMES and Its Core Resources

Free texts on or off the Web exhibit a very high degree of irregularities. Therefore, experiences in the field of Natural Language Processing have shown that complete understanding of the texts is infeasible at least in the next few years, but probably in the foreseeable future. Simultaneously, the sheer amount of texts makes it necessary to process the texts efficiently. This has lead to so called "shallow" text processing approaches, where certain generic language regularities which are known to cause complexity problems are either not handled (e.g., instead of computing all possible readings only an underspecified structure is computed) or handled very pragmatically, e.g., by restricting the depth of recursion on the basis of a corpus analysis or by making use of heuristic rules, like "longest matching substrings". This engineering view on language has lead to a renaissance and improvement of well-known efficient techniques, most notably finite state technology for parsing. Following this point of view, the major task of the core shallow text processing tools of SMES is the extraction of as much linguistic structure as possible and its mapping into — comparatively simple — conceptual structures.

In order to achieve this efficiently and robustly, SMES makes use of advanced finite state technology on all levels of processing and of a data structure called *text chart*. SMES has been fully implemented for German with high coverage on the lexical and syntactic level and with an excellent speed. We have also implemented first versions of SMES for English and Japanese using the same core technology. In this paper we will however focus on processing German text documents.

System Architecture. The architecture of SMES is shown in Figure 1. It consists of two major components, a) the Linguistic Knowledge Pool LKP and b) STP, the shallow

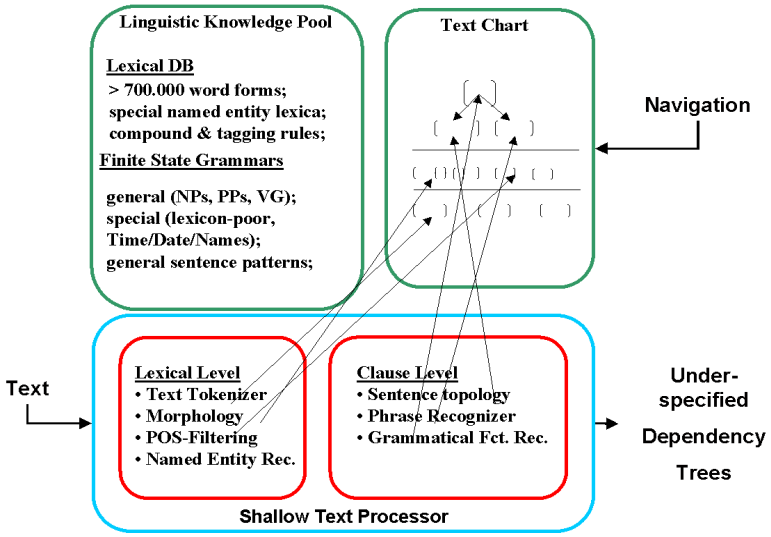

Fig. 1. The blueprint of the system architecture of the IE–core system SMES

text processor itself. STP processes a NL-text through a chain of modules. We distinguish two primarily levels of processing, the *lexical level* and the *clause level*. Both are subdivided into several components. All lexical and grammatical components of SMES are realized by means of cascaded weighted finite state machines. The final result for a sentence computed by SMES is an underspecified dependency structure, where only upper bounds for attachment and scoping of modifiers are expressed.

Tokenization. The tokenizer maps sequences of consecutive characters into larger units called tokens and identifies their types. Currently we use more than 50 token classes including generic classes for semantically ambiguous tokens (e.g., "10:15" could be a time expression or volleyball result, hence we classify this token as number-dot compound) and complex classes like abbreviations or complex compounds (e.g., "AT&T-Chief"). It proved that such variety of token classes simplifies the processing of subsequent submodules significantly.

Morphology. Each token identified as a potential wordform is submitted to the morphological analysis including on-line recognition of compounds (which is crucial since compounding is a very productive process of the German language) and hyphen coordination (e.g., in "An- und Verkauf" (*purchase and sale*) "An-" is resolved to "Ankauf" (*purchase*)). Each token recognized as a valid word form is associated with the list of its possible readings, characterized by stem, inflection information (e.g., information concerning case, person, gender) and part of speech category.

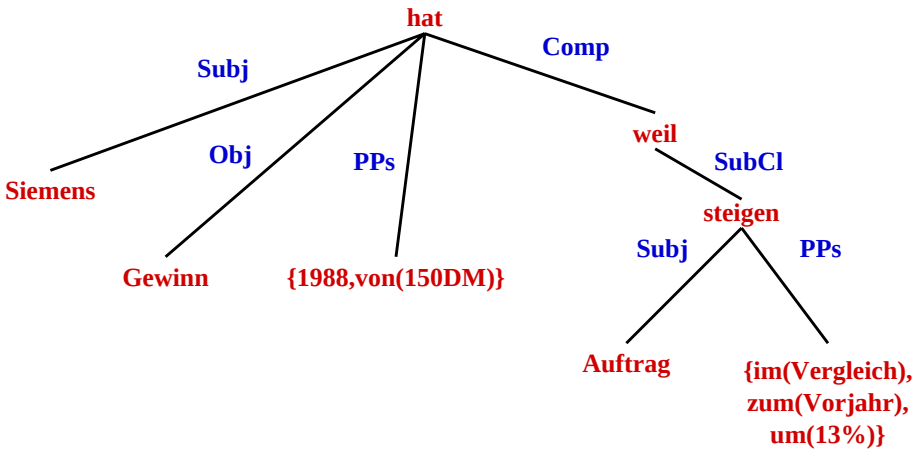

Fig. 2. The underspecified dependence structure of the sentence: "Die Siemens GmbH hat 1988 einen Gewinn von 150 Millionen DM, weil die Aufträge im Vergleich zum Vorjahr um 13% gestiegen sind" (*The Siemens company had made a revenue of 150 million marks in 1988 since the orders increased by 13% compared to last year*)

POS-filtering. Since a high amount of German word forms is ambiguous, especially word forms with verb reading[1], and due to the fact that the quality of the results of the syntactic analysis phase relies essentially on the proper recognition of verb groups, efficient disambiguation strategies are needed. Using case-sensitive rules is straightforward and reliable since generally only nouns (and proper names) are written in standard German with a capitalized initial letter (e.g., "das Unternehmen" - *the enterprise* vs. "wir unternehmen" - *we undertake*). However for disambiguation of word forms appearing at the beginning of the sentence local contextual filtering rules are applied. For instance, the rule which forbids the verb written with a capitalized initial letter to be followed by a finite verb would filter out the verb reading of the word "unternehmen" in the sentence "Unternehmen sind an Gewinnmaximierung interesiert." (*Enterprises are interested in maximizing their profits*). It is important to notice that such rules are based on some regularities, but they may yield false results, like for instance the rule for filtering out the verb reading of some word forms extremely rarely used as verbs (e.g., "recht" - *right, to rake* (3rd person,sg)). Apart from manually constructed rules mentioned above, we also use rules determined by Brill's tagger [3].[2] All rules are compiled into a single finite-state transducer according to the approach described in [20].

[1] 30% of the wordforms in the test corpus "Wirtschaftswoche" (business news journal), which have a verb reading, turned to have at least one other non-verb reading.

[2] The manually constructed rules proved to be a reliable means for disambiguation, however not sufficient enough to filter out all unplausible readings. Hence supplementary rules determined by Brill's tagger were used in order to achieve broader coverage.

Named entity finder. Named entities such as organizations, persons, locations and time expressions are dynamically identified using finite-state grammars. Since some named entities (e.g. company names) may appear in the text either with or without a designator, we use a dynamic lexicon to store recognized named entities without their designators (e.g., "Braun AG" vs. "Braun") in order to identify subsequent occurrences correctly. In other words, our named entity finder creates online a lexical database of just identified named entities, and thus performs already one (small) part of the knowledge acquisition cycle.

Chunk parsing. In most of the well-known shallow text processing systems cascaded chunk parsers are used which perform clause recognition after fragment recognition following a bottom-up style as described in [1]. We have also developed a similar bottom-up strategy for the processing of German texts, cf. [18]. However, the main problem we experienced using the bottom-up strategy was insufficient robustness: because the parser depends on the lower phrasal recognizers, its performance is heavily influenced by their respective performance. As a consequence, the parser frequently wasn't able to process structurally simple sentences, because they contained, for example, highly complex nominal phrases. For that reason we developed a novel mixed topdown/bottom–up chunk parser, which consists of three major subcomponents:

1. recognition of the topological field structure of a sentence,
2. application of phrasal grammars (e.g., nominal and prepositional phrases) to the different fields,
3. recognition of the grammatical functions (like subject, object).

During the first step of the parser a cascade of finite-state grammars are applied to the stream of lexical tokens and named entities in order to determine the topological structure of the sentence according to the linguistic field theory. Based on the fact that in German a verb group (like "hätte überredet werden müssen" — **have convinced been should* meaning *should have been convinced*) can be split into a left and a right verb part ("hätte" and "überredet werden müssen'") these parts (abbreviated as LVP and RVP) are used for the segmentation of a main sentence into several parts: the front field (VF), the left verb part, middle field (MF), right verb part, and rest field (RF). Subclauses can also be expressed in that way such that the left verb part is either empty or occupied by a relative pronoun or a subjunction element, and the complete verb group is placed in the right verb part. Note that each separated field can be arbitrarily complex with very few restrictions on the ordering of the phrases inside a field.

Grammatical function recognition. After the fragment recognizer has expanded the corresponding phrasal strings, a further analysis step is done by the *grammatical function recognizer* (GFR) which identifies possible *arguments* on the basis of the lexical subcategorization information available for the local head. The final output of the clause level for a sentence is thus an underspecified dependence structure UDS. An UDS is a flat dependence-based structure of a sentence, where only upper bounds for attachment and scoping of modifiers are expressed (see Figure 2). In this example the PPs of each main or sub-clause are collected into one set. This means that although the exact attachment point of each individual PP is not known it is guaranteed that a PP can only

be attached to phrases which are dominated by the main verb of the sentence (which is the root node of the clause's tree). However, the exact point of attachment is a matter of domain-specific knowledge and hence should be defined as part of the domain knowledge of an application.

Most of the linguistic information actually used in grammatical function recognition comes from a large subcategorization lexicon (currently counting more than 15.000 entries for German verbs). In particular, the information conveyed by the verb subcategorization lexicon we use, includes subcategorization patterns, like arity, case assigned to nominal arguments, preposition/subconjunction form for other classes of complements.

3 Ontologies

Following Tom Gruber, we understand ontologies as a formal specification of a shared conceptualization of a domain of interest to a group of users (cf. [9]). In this section we introduce ontologies by giving a mathematical definition of the elements we consider an ontology is consisting of. Thereby, an important aspect is that we use a lexicon which is mapped onto the concepts and relations of the ontology, mediating between SMES and the knowledge structures. Within our framework we use two tools: *(1)* OntoEdit that is a comprehensive ontology management system supporting the engineering of ontologies and *(2)* OntoBroker, a deductive, object-oriented inference engine that allows querying and reasoning on the defined knowledge structures.

3.1 Ontologies – A Definition

We distinguish between ontologies "in a wider sense" and "in a narrow sense". Similar to the term "information system in a wider sense", the former notion of ontologies contains many aspects, which either cannot be formalized at all or which cannot be formalized for practical reasons. For instance, it might be too cumbersome to model the interests of all the persons involved. Concerning the formal part of ontologies (or "ontology in the narrow sense"), we employ a two-part structure. The first part (cf. Definition 1) describes the structural properties that virtually all ontology languages exhibit. Note that we do not define any syntax here, but simply refer to these structures as a least common denominator for many logical languages, such as OIL [7] or F-Logic [14]:

Definition 1. *Let $\mathcal{L}$ be a logical language having a formal semantics in which inference rules can be expressed. An* abstract ontology *is a structure $\mathcal{O} := (C, \leq_C, R, \sigma, \leq_R, IR)$ consisting of*

- *two disjoint sets C and R whose elements are called* concepts *and* relations*, resp.,*
- *a partial order $\leq_C$ on C, called* concept hierarchy *or* taxonomy*,*
- *a function $\sigma\colon R \to C \times C$ called* signature*,*
- *a partial order $\leq_R$ on R where $r_1 \leq_R r_2$ implies $\sigma(r_1) \leq_{C\times C} \sigma(r_2)$, for $r_1, r_2 \in R$, called* relation hierarchy*.*
- *and a set IR of inference rules expressed in the logical language $\mathcal{L}$.*

The function $\mathrm{dom}\colon R \to C$ *with* $\mathrm{dom}(r) := \pi_1(\sigma(r))$ *gives the* domain *of* r*, and the function* $\mathrm{range}\colon R \to C$ *with* $\mathrm{range}(r) := \pi_2(\sigma(r))$ *gives its* range.

To connect the abstract ontology structure to natural language we use an explicit representation of the lexical level.[3] Therefore, we define a lexicon for our abstract ontology $\mathcal{O}$ as follows:

Definition 2. *A* lexicon *for an abstract ontology* $\mathcal{O} := (C, \leq_C, R, \sigma, \leq_R, IR)$ *is a structure* $Lex := (S_C, S_R, Ref_C, Ref_R)$ *consisting of*

- *two sets* S_C *and* S_R *whose elements are called* signs (lexical entries) for concepts *and* relations*, resp.,*
- *and two relations* $Ref_C \subseteq S_C \times C$ *and* $Ref_R \subseteq S_R \times R$ *called* lexical reference assignments for concepts/relations*, resp.*

Based on Ref_C*, we define, for* $s \in S_C$*,*

$$Ref_C(s) := \{c \in C \mid (s, c) \in Ref_C\}$$

and, for $c \in C$*,*

$$Ref_C^{-1}(c) := \{s \in S \mid (s, c) \in Ref_C\} \ .$$

Ref_R *and* Ref_R^{-1} *are defined analogously.*

The abstract ontology is made concrete through naming. Thus:

Definition 3. *A (concrete)* ontology *(in the narrow sense) is a pair* $(\mathcal{O}, Lex)$ *where* $\mathcal{O}$ *is an abstract ontology and* Lex *is a lexicon for* $\mathcal{O}$*.*

3.2 OntoEdit - Ontology Engineering Environment

OntoEdit is an ontology engineering environment supporting the ontology development and maintenance process (cf. [24]). Figure 3 depicts a screenshot of OntoEdit and a tourism ontology developed within the GETESS project [23]. On the left side the concept hierarchy of the tourism ontology is depicted. In the middle the non-taxonomic relations of the selected concept MUSEUM are depicted. On the right lower side the view of OntoEdit for adding lexical entries to concepts and relations is depicted. The definition of these mappings is an important aspect, allowing to connect the ontology structures to the natural language processing system SMES. In this concrete example morphologically reduced lexical entries are mapped onto concepts. In addition, the language of the specific lexical entry and its part of speech are defined.

OntoEdit also allows the definition of axioms about concepts and relations, like the symmetry of the "marriedWith" relation between two persons (cf. [25] for an elaborated overview). OntoEdit accesses an external inference engine that is described in the next subsection.

[3] Our distinction of lexical entry and concept is similar to the distinction of word form and synset used in WordNet [6]. WordNet has been conceived as a mixed linguistic / psychological model about how people associate words with their meaning.

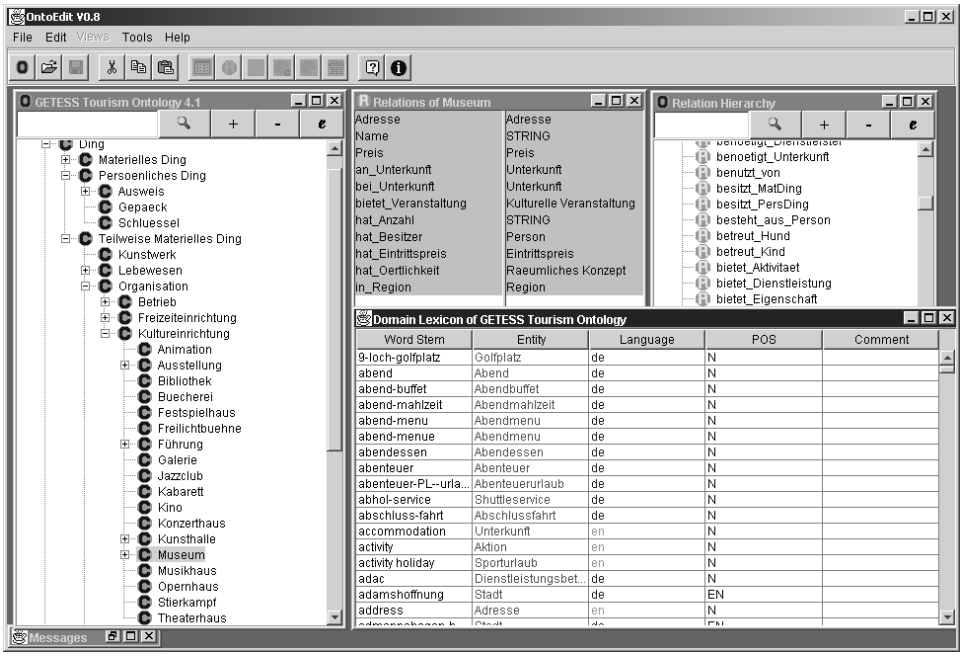

Fig. 3. OntoEdit Ontology Engineering Environment

3.3 Inference Engine

We use the Ontobroker system [5] as a component of our information extraction system. Ontobroker is a deductive, object-oriented database system. It is based on F-Logic [14] allowing to describe and instantiate an ontology.

Let us consider a short example of how to use OntoBroker with its underlying representation language F-Logic. Concept definitions in F-Logic for ACCOMODATION having a relation LOCATEDIN with range AREA and HOTEL being a subconcept of ACCOMODATION is given in (1), while a fact that 'Schwarzer Adler' is a HOTEL which is LOCATEDIN Rostock appears like in (2).

(1) ACCOMODATION[LOCATEDIN $\Rightarrow\!\!>$ AREA].
HOTEL::ACCOMODATION.

(2) 'Schwarzer Adler':HOTEL[LOCATEDIN $\rightarrow\!\!>$ 'Rostock'].

Hence, an inference rule describing that the relation LOCATEDIN is transitive is given by (3).

(3) FORALL X, Y, Z $X[\text{LOCATEDIN} \rightarrow\!\!> Z] \leftarrow X[\text{LOCATEDIN} \rightarrow\!\!> Y]$
AND $Y[\text{LOCATEDIN} \rightarrow\!\!> Z]$.

Ontobroker allows querying as well on the schema as on the instance level, e.g., the following query retrieves all domain / range pairs of all non-taxonomic relations defined in the ontology:

(4) FORALL $X, Y \;\; X, Y \leftarrow$ EXISTS $R \;\; X[R \twoheadrightarrow Y]$.

4 Bridging

In human understanding of free text, syntax, semantics, context, and/or knowledge may trigger the search for conceptual bridges between syntactic objects (cf. [21]). For instance,

- **Syntax**: the syntactic dependencies in the phrase "the house of Usher" signal a conceptual dependency between the conceptual denotations corresponding to "house" and "Usher".
- **Semantics**: In the phrase "The baby have cried." the semantic restrictions allow to infer the conceptual relationship between the denotates of "baby" and the "cry"ing — even though the sentence is syntactically illegitimate.
- **Context**: In "They are geniuses. Michael, Dieter, Markus." the resolution of "Michael being a genius, etc." may occur because of contextual cues (and ellipsis resolution) (cf. e.g., [17]).
- **Knowledge**: In "CPU A is faster than B.", knowledge is responsible to associate the denotates of cpu A and B with a comparison of their frequency figures rather than their physical speed (because they could be traveling in a vehicle).

SMES constitutes the IE system component for signaling syntactic cues. In IE one typically only considers syntax to trigger the search for a conceptual bridge. Thereby, one uses the background knowledge given in the ontology to check for the possibility of conceptual bridges. In the easiest case the ontology is directly used for validating a conceptual bridge, e.g., 'Mecklenburg-Vorpommern in Eastern Germany' corresponds to the ontology structure REGION CONTAINS REGION. Therefore, the conceptual bridge between 'Mecklenburg-Vorpommern' and 'Eastern Germany' may be positively validated.

However, there exists a wide range of possibilities according to which a bridge may be built given a particular ontology. The principal variance comes from effects such as granularity, metonomy, or figurative language. For instance, one may model in the ontology that a country contains states and states contain counties. Because of the transitivity of the contains relationship, the ontology also allows the direct connection of country with county.

In our concrete implementation we use OntoBroker to check for valid conceptual bridges. Thereby, we focus on syntactically motivated ones. We ignore metonymic and figurative language, because they currently constitute research topics of their very own.

We want to mention that extraction rules may be considered as implementations of ideosyncratic syntactic cues for particular ontological structures. For example, we have particular extraction rules for addresses, company names, etc.

5 Bootstrapping

A general problem of information extraction is that it requires large conceptual and linguistic resources. Therefore, we apply a bootstrapping mechanism to semi-automatically

acquire the required resources. In the following we first describe our mechanism for the extraction of knowledge structures from natural language texts. Subsequently, we extend this approach for the extension and adaptation of linguistic resources. Finally, we analyze their interaction in the bootstrapping cycle.

5.1 Ontology Learning

Ontology Learning is a mechanism that semi-automatically supports the process of extracting and maintaining an ontology from a given set of data [4]. Thus, it applies machine learning techniques on given data for the automatic extraction of knowledge structures that are proposed to and refined by the ontology engineer.

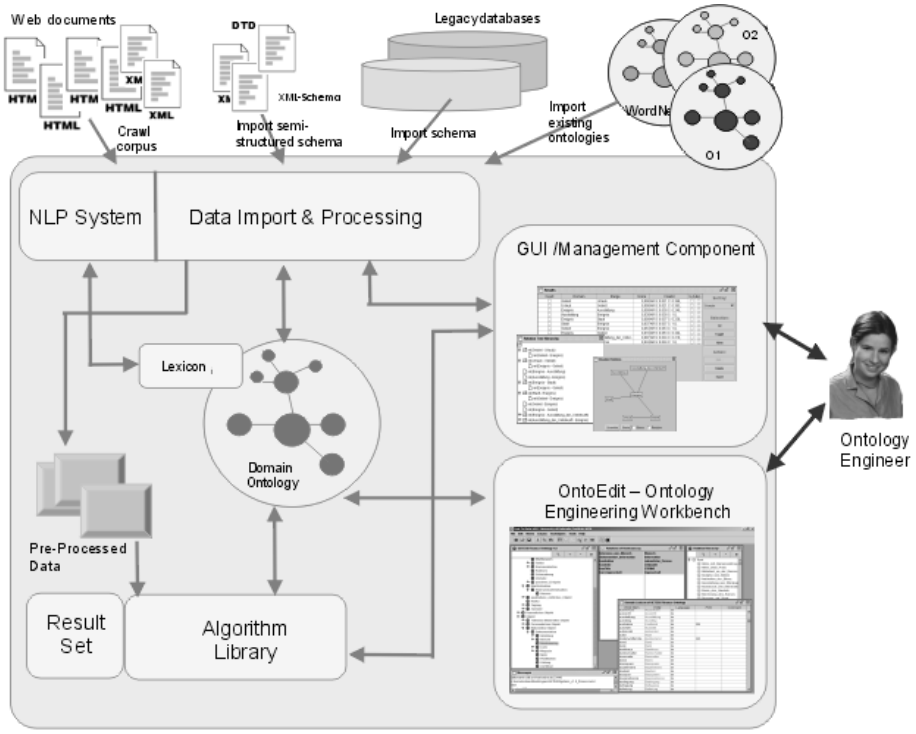

Fig. 4. Architecture for Ontology Learning

We here only give a rough outline of the Ontology Learning approach by introducing our generic architecture for extracting and maintaining ontologies from given data, in particular natural language texts. We have identified four main core architectural components. There are,

[4] A comprehensive overview and introduction into Ontology Learning is given in [16].

– A generic **management component** dealing with delegation of tasks and constituting the infrastructure backbone,
– A **data import and processing component** working on input data from the Web including, in particular, the natural language processing core component SMES,
– An **algorithm library**[5] working on the output of the data import & processing component as well as the ontology structures sketched above and returning result sets also mentioned above and,
– The **ontology management and engineering environment** OntoEdit.

The interaction between these components is depicted graphically in Figure 4. An important aspect is that the data import and processing component as well as the algorithm library component access the available knowledge structures in each step.

5.2 Interaction

Ontology Learning is one important step in the overall bootstrapping approach we pursue. The overall system uses an incremental bootstrapping approach in the following sense:

1. We start with a shallow easily reusable IE model that is given as a baseline with the core natural language system SMES (see Section 2)
2. Then, a domain specific corpus is selected.
3. The corpus is processed with the core IE system (exploiting the given IE model);
4. Based on this data one uses a set of different learning approaches (from very simple statistical approaches, like n–grams up to complex ones like ILP) embedded into the Ontology Learning framework.
5. The IE model is extended. In particular, we support the extension of the lexical knowledge, extraction rules, and the ontology. Here comes the human into the loop in order to review learning decisions.
6. We continue with step 3. until the human modeler decides to stop.

6 Related Work

We distinguish our approach along two dimensions. First, as a response to problems that knowledge representation suffered from until the early 90's, people in information extraction built IE models with simplified knowledge structures, namely templates, which came with little inferencing capabilities[6]. In general, templates provide a flat knowledge structure that is to be instantiated with attribute values. The prototypical architecture for a template-based information extraction system has been introduced in [2]. Because, there are now efficient inferencing systems, the ontology-based description of the domain should be favored over the comparatively rigid template-based specifications.

[5] The interested reader is referred to [15, 13]

[6] Some proposals make use of type-based feature structures and unification, but they do not provide an easily understandable conceptual model of the domain.

Second, work on the explicit combination of information extraction and machine learning has mostly been focused on only few parts of the IE model, e.g. described by [12, 19, 22, 4, 8]. The underlying idea is rather than spending weeks or months manually adapting an information extraction system to a new domain, one would like a system that can be trained on some sample documents and that is expected to do a reasonable job of extracting information from new ones. In [8] a multi-strategy learning architecture for the generation of extraction patterns is introduced. Along the same lines, [26] present an automatic discovery procedure called ExDisco which identifies a set of event patterns from un-annotated text, starting from a small set of seed patterns. Their approach shows a significant performance improvement on actual extraction tasks in contrast to manually constructed systems.

There are some systems that target a knowledge-oriented text extraction and understanding strategy with a tight integration between ontology, lexicon and extraction rules and an incremental refinement of the IE model. In [10] the Syndicate system that targets the transformation of documents into a knowledge base is introduced. In contrast to our approach the ontology learning part of Syndicate restricts its attention to the techniques for ontology refinement.

7 Discussion

In this paper we have presented a comprehensive approach for bootstrapping an ontology-based information extraction system with the help of machine learning. The proposed architecture has been instantiated within the GETESS (German Text Exploitation and Search System) project (see [23]), where domain-oriented search services for tourism and finance information have been developed. The further application of our approach will appear in the Semantic Web, e.g. in the area of semantic annotation and metadata generation (see [11]).

Acknowledgements. The research presented in this paper has been partially funded by BMBF under grant number 01IN802 (project “GETESS”) and by US Air Force in the DARPA DAML project “OntoAgents” (01IN901C0).

References

1. S. Abney. Partial parsing via finite-state cascades. Proceedings of the ESSLLI 96 Robust Parsing Workshop, 1996.
2. D. Appelt, J. Hobbs, J. Bear, D. Israel, and M. Tyson. Fastus: A finite state processor for information extraction from real world text. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Chambery, France, August 1993.
3. E. Brill. Automatic grammar induction and parsing free text: A transformation-based approach. In *31th Annual Meeting of the Association for Computational Linguistics*, Ohio, 1993.
4. M. Califf and R. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the AAAI Spring Symposium on Applying Machine Learning to Discourse Processing*, 1998.

5. S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In R. Meersman et al., editors, *Database Semantics: Semantic Issues in Multimedia Systems*, pages 351–369. Kluwer Academic Publisher, 1999.
6. Christiane Fellbaum. *WordNet – An electronic lexical database*. MIT Press, Cambridge, Massachusetts and London, England, 1998.
7. D. Fensel, F. van Harmelen, I. Horrocks, D. L. McGuinness, and P. F. Patel-Schneider. OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–44, March/April 2001.
8. D. Freitag. *Machine Learning for Information Extraction in Informal Domains*. PhD thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA,USA, 1998.
9. T. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220, 1993.
10. U. Hahn and M. Romacker. Content management in the SYNDIKATE system - How technical documents are automatically transformed to text knowledge bases. *Data & Knowledge Engineering*, 35(2):137–159, 2000.
11. S. Handschuh, S. Staab, and A. Maedche. CREAM — Creating relational metadata with a component-based, ontology driven framework. In *Proceedings of the First ACM Conference on Knowledge Capture – K-CAP'01*, 2001.
12. S. Huffman. Learning information extraction patterns from examples. In Wermter, Riloff, and Scheler, editors, *Connectionist, Statistical, And Symbol Approaches to Learning for Natural Language Processing*, volume 1040 of *Lecture Notes in Artificial Intelligence*, pages 246–260, Berlin, Springer, 1996.
13. J.-U. Kietz, R. Volz, and A. Maedche. A method for semi-automatic ontology acquisition from a corporate intranet. In *EKAW-2000 Workshop "Ontologies and Text", Juan-Les-Pins, France, October 2000.*, 2000.
14. M. Kifer, G. Lausen, and J. Wu. Logical Foundations of Object-Oriented and Frame-Based Languages. *Journal of the ACM*, 42(4):741–843, 1995.
15. A. Maedche and S. Staab. Discovering conceptual relations from text. In *ECAI-2000 - European Conference on Artificial Intelligence. Proceedings of the 13th European Conference on Artificial Intelligence*. IOS Press, Amsterdam, 2000.
16. A. Maedche and S. Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16(2), 2001.
17. K. Markert and U. Hahn. On the interaction of metonymies and anaphora. In *Proc. of IJCAI-97*, pages 1010–1015, 1997.
18. G. Neumann, R. Backofen, J. Baur, M. Becker, and C. Braun. An information extraction core system for real world german text processing. In *5th International Conference of Applied Natural Language*, pages 208–215, Washington, USA, March 1997.
19. E. Riloff. Using learned extraction patterns for text classification. In Wermter, Riloff, and Scheler, editors, *Connectionist, Statistical, And Symbol Approaches to Learning for Natural Language Processing*, volume 1040 of *Lecture Notes in Artificial Intelligence*, pages 275–289, Berlin, Springer, 1996.
20. E. Roche and Y. Schabes. Deterministic part-of-speech tagging with finite state transducers. *Computational Linguistics*, 21(2):227–253, 1995.
21. M. Romacker, K. Markert, and U. Hahn. Lean semantic interpretation. In *Proc. of IJCAI-99*, pages 868–875, 1999.
22. S. Soderland. *Learning Text Analysis Rules for Domain Specific Natural Language Processing*. PhD thesis, University of Massachusetts Amherst, 1997.
23. S. Staab, C. Braun, A. Düsterhöft, A. Heuer, M. Klettke, S. Melzig, G. Neumann, B. Prager, J. Pretzel, H.-P. Schnurr, R. Studer, H. Uszkoreit, and B. Wrenger. GETESS — searching the

web exploiting german texts. In *CIA'99 — Proceedings of the 3rd Workshop on Cooperative Information Agents. Upsala, Sweden, July 31-August 2, 1999*, LNCS 1652, pages 113–124, Berlin, 1999. Springer.

24. S. Staab and A. Maedche. Ontology engineering beyond the modeling of concepts and relations. In V.R. Benjamins, A. Gomez-Perez, and N. Guarino, editors, *Proceedings of the ECAI-2000 Workshop on Ontologies and Problem-Solving Methods. Berlin, August 21-22, 2000*, 2000.
25. Steffen Staab, M. Erdmann, and A. Maedche. From manual to semi-automatic semantic annotation: About ontology-based text annotation tools. *ETAI – Semantic Web Journal, Linkoeping Electronic Articles*, 16(1), 2001.
26. R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. Automatic Acquisition of Domain Knowledge for Information Extraction. In *Proceedings of COLING'2000, Saarbrcken, Germany*, 2000.