Gupta, K.M., & Aha, D.W. (2004). Towards acquiring case indexing taxonomies from text. *Proceedings of the Seventeenth International FLAIRS Conference.* Miami Beach, FL: AAAI Press.

Towards Acquiring Case Indexing Taxonomies From Text

Kalyan Moy Gupta[1, 2] and David W. Aha[2]

[1]ITT Industries; AES Division; Alexandria, VA 22303
[2]Navy Center for Applied Research in Artificial Intelligence;
Naval Research Laboratory (Code 5515); Washington, DC 20375
surname@aic.nrl.navy.mil

Abstract

Taxonomic case-based reasoning is a conversational case-based reasoning methodology that employs feature subsumption taxonomies for incremental case retrieval. Although this approach has several benefits over standard retrieval approaches, methods for automatically acquiring these taxonomies from text documents do not exist, which limits its widespread implementation. To accelerate and simplify feature acquisition and case indexing, we introduce FACIT, a domain independent framework that combines deep natural language processing techniques and generative lexicons to semi-automatically acquire case indexing taxonomies from text documents. FACIT employs a novel method to generate a logical form representation of text, and uses it to automatically extract and organize features. In contrast to standard *information* extraction approaches, FACIT's *knowledge* extraction approach should be more accurate and robust to syntactic variations in text sources due to its use of logical forms. We detail FACIT and its implementation status.

1. Introduction

Case-based reasoning (CBR) is a general methodology for retrieving and reusing past experience to solve similar new decision problems (Aamodt and Plaza 1994). Conversational CBR (CCBR) is a CBR methodology that engages a user in a question answer dialog to retrieve cases (Aha et al. 2001). It has been successfully deployed in many help-desk and troubleshooting applications. Taxonomic CBR enhances CCBR by exploiting features organized into taxonomies to shorten user adaptive conversations and improve case retrieval performance (Gupta 2001; Gupta et al. 2002).

A key challenge for applying CBR is acquiring cases from text documents (e.g., manuals, reports, logs), which is a focus of Textual CBR (Ashley and Lenz 1998). For each case, these systems must determine, or be told, which of the predefined features to use as indices. When the mapping of indexing features to text cases is simple, a bag-of-words approach can be used to automate feature extraction (e.g., Burke et al. 1997). However, when this mapping is complex, these features must be manually identified (e.g., Weber et al. 1998; Brüninghaus and Ashley 2001). Unfortunately, this approach fails when the features are not known a priori, as is true for complex troubleshooting applications and many other domains. To our knowledge, this feature acquisition problem has not been addressed previously. The problem is further compounded by Taxonomic CBR's need to organize features into subsumption taxonomies (Gupta 2001). Fortunately, this feature acquisition and organization problem can be addressed by *knowledge extraction* techniques (Cowie and Lehnert 1996), which aim to deduce knowledge artifacts such as rules, cases, and domain models from text. However, knowledge extraction involves significantly more complex natural language processing (NLP) methods than do traditional information extraction (IE) approaches.

In this paper, we introduce knowledge extraction methodologies in the form of a domain independent framework for feature acquisition and case indexing from text (FACIT). FACIT uses deep NLP techniques with a generative lexicon for semantic interpretation to enable robust interpretation of previously unseen text documents (Gupta and Aha 2003). In a forthcoming paper, we present evidence that standard IE techniques perform poorly in comparison to FACIT's knowledge extraction techniques on this feature organization task (Gupta et al. 2004).

We next describe related work on acquiring case indices from text. We then introduce FACIT, illustrating its processes with an example. Finally, we report on FACIT's implementation status and discuss future research ideas.

2. Methods for Indexing Text Cases

Several engineering processes for acquiring high quality cases exist (e.g., Gupta 1997). Developers typically consult documented knowledge and subject matter experts (e.g., for an equipment diagnosis application, they may rely on troubleshooting manuals, maintenance logs, and failure modes analysis). Developers select the case material, translate it into language for end user consumption, and encode it into a representation for use by the CBR system.

Unfortunately, these processes are mostly manual, are minimally supported by editors, and require a significant amount of skill, effort, and time. This complicates the wide spread application of CBR, and is exacerbated by the use of increasingly sophisticated CBR methodological variants such as Taxonomic CBR. Thus, case index acquisition can be significantly accelerated by using software tools that assist with identifying, extracting, and transforming content from text sources to identify their indices.

Several researchers have developed methods for *assigning* indices to text documents, as summarized in Table 1, but none appear to have developed index extraction methods for domain-independent, unstructured

Table 1: Characterizing approaches for assigning indices to text documents. Legend: **AV**=Attribute-value; **CRN**=Case retrieval network; **DSF**=Developer supplies features; **DSL**=Developer supplies lexicon; **DSS**=Developer supplies similarity information; **ESV**=Expert supplies values (for attributes); **EV**=Expert validation); **FT**=Free text; **IEs**=Information-entity pairs; **ProPs**=Propositional patterns; **SE**=Sense enumerative; **SST**=Semi-structured text

Name + Reference	Source	Lexicon & Representation	Patterns	Concept Representation	Case Representation	Human Roles
FAQ Finder (Burke *et al*., 1997)	SST	SE	None	WordNet Synset #	Sense-Tagged Bag of Words	DSL
Prudentia (Weber *et al*., 1998)	SST	None	Templates	None	AV	DSF
FAllQ (Lenz *et al*., 1998)	SST	SE	IEs	WordNet Synset #	CRN	DSL, DSF, DSS
Indexing Assistant (Baudin & Waterman, 1998)	FT	Concept Taxonomy	None	None	AV	DSL, DSF, ESV
(Minor & Hübner, 1999)	SST	SE	IEs	Dictionary Terms	CRN	DSL, DSF
SMILE (Brüninghaus & Ashley, 1999)	FT	SE	Induced classifier	Bag of Words	AV (Factors)	DSL, DSF, ESV
SMILE+AutoSlog (Brüninghaus & Ashley, 200l)	FT	SE	Induced Rules and ProPs	ProPs	AV (Factors)	DSL, DSF, ESV
IDS+ (Yang *et al.*, 2003)	SST	None	Templates	None	AV	DSL, DSF, EV
FACIT (Gupta & Aha, this paper)	**FT**	**Generative**	**None**	**Logical Form**	**Taxonomic**	**DSL, EV**

text documents. *Source* refers to the source documents. While most methods use lexicons that are *sense enumerative,* FACIT's lexicon is *generative* (see Section 3). Several methods use patterns/templates to assign indexing features, while more sophisticated methods (e.g., SMILE) automatically construct these patterns. In contrast, FACIT does *not* use patterns to extract features. Most methods use a bag-of-words approach to represent concepts in the text. An exception is propositional patterns (Brüninghaus and Ashley 2001), which provide an abstraction mechanism. In contrast, FACIT uses canonical logical forms to represent text that may contain features; this enables reasoning on the information content of the source documents. Most methods use an attribute-value (*feature*) index representation, such as a set of fields in a template, although case retrieval networks (Lenz et al. 1998) have also been used. In contrast, FACIT derives feature subsumption taxonomies in addition to features. Invariably, developers and/or experts serve multiple roles during the feature assignment process. This is also true for FACIT, but to a much lesser degree; we assume a domain expert will provide feedback only on whether sampled sentences contain features of interest. In summary, FACIT is the first index acquisition methodology to use a generative semantic lexicon and a logical form representation for extracted features. This permits it to identify feature relations, and thus generate feature subsumption taxonomies.

3. Acquisition and Indexing Framework

FACIT updates a semantic lexicon and uses it for syntactic and deep semantic interpretation to create a complete and valid logical form representation, which is a set of sentences represented in a predicate argument structure. FACIT extracts features from the logical form to index cases. We next describe and illustrate FACIT's six steps (see Figure 1) by processing example sentences from the troubleshooting chapter of Canon's Multipass C-5500 printer user manual (CSSI 1998). Steps 2-4 implement a knowledge extraction process.

1. Update the semantic lexicon: NLP systems employ *lexical knowledge bases* (LKBs) to look up potential senses (*concepts*) associated with a term and use a disambiguation technique to select the most applicable sense among them. Domain independent LKBs (e.g., WordNet (Felbaum 1998), Sensus (Knight and Luk 1994)) have poor coverage for domain specific text applications. For example, WordNet covers only 25.6% of the terms from our naval training exercises domain (Gupta et al. 2002). Its coverage of senses is likely to be even lower because it lacks domain specific senses for known terms. Consequently, selected lexical resources must include domain specific terms and senses. Thus, issues of concern include the lexicon choice and the effort required to update it.

Semantic lexicons can be categorized as either *sense enumerative* (e.g., WordNet, Sensus) or *generative* (Pustejovsky 1995). Enumerative lexicons, which require listing every sense of a term or phrase in the lexicon, have weak lexical semantics (few impoverished relation types between concepts), weak compositionality (cannot derive the meaning of an unlisted phrase from its constituent terms), and large sense ambiguities. Thus, the effort to update such lexicons increases linearly with the number of unknown terms and phrases. In contrast, generative lexicons (GLs) include rich, well-principled semantics (can express an unlimited set of relations) and do not require

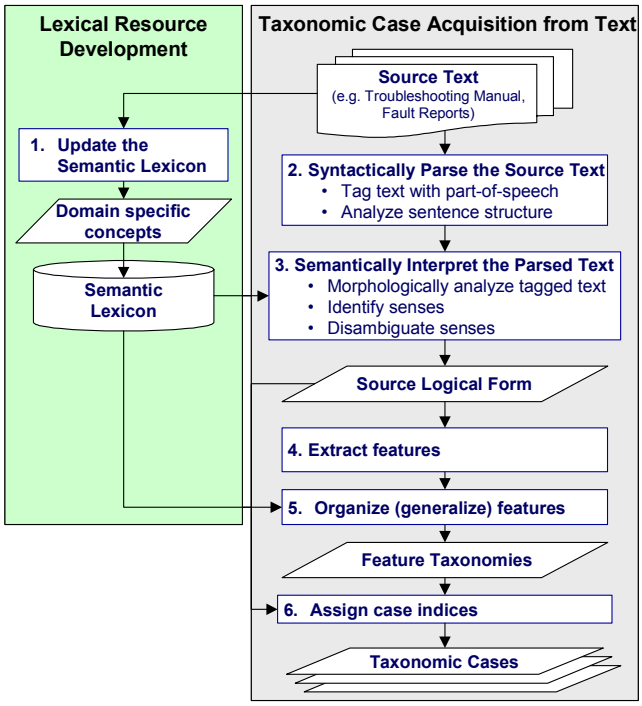

Figure 1: The FACIT framework processes and steps

explicitly listing all potential senses of a term. Instead, a small set of powerful operators generates them on demand from their context of use. GL supports strong compositionality and can derive senses of previously unseen term combinations. The effort required to update GLs is sublinear and comparatively marginal.

We developed several extensions to GL theory and implemented these in a representation called *Sublanguage Ontology* (SO) (Gupta and Aha 2003). We also developed software tools that support the development and maintenance of SOs, including the Sublanguage Ontology Editor, which allows users to edit new and existing concepts, and the Concept Discovery Workbench (CDW), which supports the semi-automatic acquisition of concepts from text documents. This greatly simplifies and accelerates ontology updating.

Using the CDW to discover terms from the Multipass C-5500 manual reveals new terms and concepts for updating the lexicon. For example, these include noun terms and phrases such as *Multipass C-5000*, *sheet feeder or automatic document feeder*, *power cord*, and *interface cable*. Also, compound terms like *sheet feeder* can be compositionally interpreted as an INSTRUMENT/PART for feeding sheets. CDW's output is used to (manually) create new concepts, as exemplified in Figure 2, which can be related to existing concepts within the SO.

SOs can represent considerable lexical and domain knowledge using inheritable objects and relations. For example, Figure 2 shows that MULTIPASS-C-5500 is a type of PRINTING_INSTRUMENT that human agents use to print INFORMATION and is an artifact made by the organization CANON_INC. It includes PARTs such as

MULTIPASS-C-5500 <
terms: <Multipass-C-5500/n, unit/n>
type_of: <PRINTING_INSTR>
attributes: <
!SIZE(this,unspecified)
!COLOR(this, unspecified)…>
constituents: <
PART(this, SHEETFEEDER),
PART(this, OP_PANEL),
PART(this, POWER_CORD),…>
behaviors <
!PRINT_ACT(HUMAN,INFO, this)>
MOVE_ACT(HUMAN,INFO,…)>
creative events <
MAKE_ACT(CANON_INC, this)>>

SHEETFEEDER <
terms <sheet feeder/n, ADF/n>
type_of: <FEEDING_INSTR>
attributes: <
!SIZE(this, unspecified)
!COLOR(this, unspecified),… >
constituents: <
PART(this, PAPER_GUIDE),…>
behaviors <
FEED_ACT (HUMAN,PAPER, this, MULTIPASS-C-5500)>
creative events <
MAKE_ACT(CANON_INC,this)>>

Legend: CONCEPTS; ***Slotnames***; <values>; **!**inherited slot; /part of speech (e.g.,n=Noun)

Figure 2: Two MULTIPASS sublanguage ontology concepts

SHEETFEEDER, which in turn includes a part PAPER_GUIDE. The domain knowledge acquired during this phase will be the basis for feature organization (step 5). In addition, the noun term *unit* was added as a synonym for MULTIPASS. Other senses of *unit* that are irrelevant to the selected application can be suppressed to prevent unnecessary ambiguity resolution overhead in step 3. In technical domains, nouns representing components, parts, and names account for the majority of lexicon updates.

2. Syntactically parse the source text: Although case extraction methods often assume each document contains a case, this depends on the document type. For example, troubleshooting manuals often contain information in a tabular format[1] from which cases must be constructed prior to encoding. This preprocessing step, not shown in Figure 1, must be performed prior to syntactic parsing. Thus, we developed the Document Extraction Workbench to help manually extract cases into arbitrarily complex structures. For example, it can be used to create a case with fields corresponding to a source document's column headings. Figure 3 displays a preprocessed input for FACIT.

Problem:	Data from the computer is not printed
Cause:	The print head unit may need cleaning.
Solution:	Clean the print head. See page 9-8.

Figure 3: Example semi-structured case text from the Multipass domain document (CSSI 1998).

Transforming text into its logical form involves a two-step process that includes syntactic parsing and semantic interpretation. Syntactic parsing assigns part-of-speech and sentence structure using a grammar and a lexicon. The sentence structure represents the grammatical structure comprising a hierarchical relationship between the terms and phrases of a sentence. For example, Figure 4 shows the parse of the sentence "Data from computer is not printed".

[1] Although this example has implicit (tabular) structure, FACIT does not require any implicit structure to extract case indices.

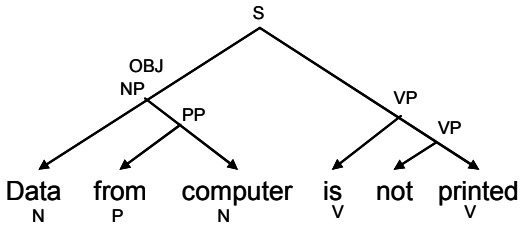

Figure 4: A syntactic parse for a printing domain sentence

Syntactic parsers are categorized as either *shallow* or *deep*. Deep parsers search for and enumerate all potential parses based on the grammar and lexicon (e.g., CMU's Link Parser (Link 2003)). Depending on a sentence's length and complexity, it may have thousands of parses, which can yield considerable sentence structure ambiguity that must be resolved by semantic interpretation. Generating all parses and selecting a valid parse among them can be computationally expensive. However, all potential parses must be considered to ensure that the valid parse will be found.

Shallow parsers use statistical, memory-based (e.g. Zavrel and Daelemans 1999), and/or data-based techniques to efficiently return one or a few top ranked parses, but they return only constituent phrases and a partial syntactic structure. This fast technique has been used in information retrieval and IE applications, whose needs can be met by a shallow parse output. However, using shallow parsing for feature extraction and assignment is problematic because:

- The likelihood of finding a valid parse can be unacceptably low.
- It shifts and increases the burden of knowledge engineering to the development of IE patterns, which provide limited domain knowledge and cannot be effectively reused to aid similarity assessment.

Because case index acquisition can be an off-line process in our domains, we use deep parsing. Furthermore, FACIT *eliminates* the use of domain specific IE patterns. As shown later, the domain knowledge acquired and stored in LKBs such as an SO can be effectively reused for similarity assessment. To this end, we have adapted the Link Parser (Link 2003) to perform deep parsing. It degrades gracefully when presented with ill-formed text by allowing broken links or structures.

3. Semantically interpret the text: Semantic interpretation transforms the grammatical form (or the syntactic parse) into a logical form, which uses predicate argument structures to represent the meaning of sentences contained in the text as propositions (see Figure 5).

INSTANCE_OF(DATA, data_1) AND
INSTANCE_OF(COMP_INSTR, computer_1) AND
NOT(PRINTED(HUMAN, data_1, PRINTING_INST)) AND
NATIVE_OF(computer_1, data_1)

Figure 5: Logical form derived from the parse in Figure 4

Sentences with different grammatical structure but the same meaning must have, or must be reducible to, the same logical form. For example, the following sentences would yield the same logical form, or meaning, as the sentence in Figure 4 except for new variable instantiations.

- Data sent from the printer to the computer is not printed
- Data is not printed by the printer
- Multipass is not printing data from the computer

A large amount of surface syntactical variation of NL text is eliminated by transforming it into its logical form. Furthermore, predicate calculus operations are applicable to logical forms, thereby enabling symbolic reasoning. We propose to use these operators for generalizing and selecting features in step 5 of FACIT.

FACIT creates the logical form from the syntactic parse as follows:

1. *Look up senses in the lexicon*: All concepts indexed by the terms in the sentence are retrieved from the LKB. For the sentence in Figure 4, our approach retrieves the concepts DATA, COMP_INSTR, and NOT for the terms *data(n)*, *computer(n),* and *not(d)*. Also retrieved are OCCUR_LOCATED and NATIVE_OF for the term *from(p)*, and PRINT_EVENT for the term *print(v)*, which was obtained by morphologically parsing the term *printed*(v). Clearly, concepts represented using predicate argument relations are necessary for deriving logical forms. Therefore, LKBs that do not support such representations cannot be directly used.
2. *Resolve semantic ambiguity:* Semantic ambiguity results when multiple concepts are retrieved for a term. Heuristics can be used to resolve these ambiguities. For example, OCCUR_LOCATED and NATIVE_OF are both retrieved for the term *from(p)*. In this case, heuristics select the concept NATIVE_OF because a larger proportion of its arguments are instantiated.
3. *Resolve syntactic ambiguity*: When multiple parses are semantically interpreted, the instantiation and predicate argument binding differ among them. FACIT selects the parse(s) that has the most predicate argument bindings as the valid one. Therefore, syntactic ambiguity resolution takes place during the semantic interpretation step.

We implemented a preliminary version of a semantic interpreter that operates with our SO and the output of the Link Parser.

4. Extract features from the logical form: Case features in a troubleshooting application are abnormal states and/or observations pertaining to a piece of equipment. For example, statements such as "Data from the computer is not printed", "Printout curls", and "Printout does not match the paper size" are abnormal conditions in a printer troubleshooting domain, whereas the statement "Make sure the computer and the application are configured correctly" is a repair instruction.

After step 3, all statements from the text are available in logical form. We propose to induce a classifier to extract features represented in logical form. To do this, we will obtain training data by asking a user to classify sampled sentences as features or non-features from a given text document. The predicates and arguments in the

corresponding logical forms will serve as the features, and we will select an appropriate learning algorithm through an analysis of the learning task. We will then use the trained classifier to extract features from all the text. Our proposed approach differs from SMILE (Brüninghaus and Ashley 1999) in that FACIT performs feature extraction and case indexing while SMILE performs only the latter. Also, FACIT's indexing strategy differs greatly (see step 6).

FACIT's feature extraction method differs from IE approaches that use patterns/templates (e.g., Weber et al. 1998) or induction to extract cases from text. Rather than using a large library of domain specific IE patterns, FACIT uses one or more classifiers to extract features. Furthermore, as we illustrate in step 5, features extracted by IE techniques do not lend themselves to generalization. Thus, FACIT requires less feature engineering, and will likely yield systems that have higher recall and precision performance than shallow NLP approaches in situations where the characterizing features are not known a priori.

5. Feature organization/generalization: To organize features into subsumption taxonomies, we propose the following procedure. First, it will perform a pair-wise comparison of each feature to examine a potential subsumption relation (i.e., a member or subset relation between two logical sentences (Russell and Norvig 1995)). These features are expressed in logical forms that include concepts and relations from the SO. We assume that each feature is a complex sentence of literals connected by connectives, as shown in Figure 5. Initially, we will consider only conjunctive expressions. The background knowledge to induce taxonomies is in the SO, which uses three types of lexical relations to assess subsumption:

1. *Is_a_type_of*: This standard relation is the basis for multiple-inheritance in a SO and applies to both entities and events. For example, the entity PRINTING_INSTR *is_a_type_of* INSTRUMENT and the event TURN *is_a_type_of* MOVE.
2. *Constituent*: This includes a family of *is_a_part_of* relations that applies to entities such as INCLUDE, WHOLE_PART, and SET_MEMBER.
3. *Is_a_subevent*: This includes hierarchical and temporal relationships between events. For example, two subevents of PRINT_ACT are FEED_PAPER_ACT, and MOVE_PRINTHEAD_ACT.

We will use additional background knowledge as needed to assess subsumption relations. For example, domain specific implication rules such as

$$\text{NOT(EVENT)} \Rightarrow \text{PROBLEM(EVENT)}$$

implies that, if an event does not occur, then there is a problem with the event. This permits the conclusion that, for example, the statement "printing Problem" subsumes the statement "Data from computer is not printed". To assess this subsumption relation, our procedure will generalize the logical form of the statement "Data from computer is not printed" by reducing the conjuncts to NOT(PRINTED(HUMAN,DATA,PRINTING_INST)) and then applying the rule to obtain the further generalization PROBLEM(PRINT_EVENT), which is a logical form for the statement "Printing problem". We will address where and how such background knowledge will be acquired and stored in our future research efforts.

After all potential subsumption relations are identified in a matrix, directed graphs, each representing a taxonomy, shall be automatically constructed and presented to the domain expert for verification.

6. Assigning indices to cases: Indexing a taxonomic case involves assigning one or more leaves from distinct feature taxonomies. Step 4 provides the logical form of features applicable to the cases. Using the feature taxonomies as a reference, FACIT will select only the most specific distinct features applicable to a case to encode it. If a most specific feature in the case is not a leaf from one of the taxonomies, then the case shall be brought to a domain expert's attention for review and correction. This process of case indexing significantly differs from those that assign predefined features (e.g., Weber et al. 1998; Brüninghaus and Ashley 1999; 2001).

4. Implementation Status

In this section, we summarize the status of the Java tools we have implemented for each of FACIT's steps.

1. *Updating a semantic lexicon*: We implemented the SO representation, the Sublanguage Ontology Editor, and the CDW to populate and update sublanguage ontologies. An SO implements a lexical representation in an XML structured repository that includes our extensions to GL theory (Gupta and Aha 2003). We are currently extending SOs to support both syntactic and semantic morphological processing that will increase the robustness of semantic interpretation.

CDW processes English text to help a knowledge engineer update SOs with domain specific terms and their related concepts. CDW discovers concept elements such as the terms, phrases, acronyms, and abbreviations for indexing SO concepts. CDW currently operates in a standalone mode. We will integrate these tools to increase the efficiency of SO updating tasks.

2. *Syntactic parser*. We implemented JLink, an interface to CMU's easily available Link Parser (Link 2003), which supports syntactic parsing. We integrated this with our Java implementation of Brill's (1995) part-of-speech tagger to help efficiently select better parses. We may later replace this with a suitable probabilistic parser.

3. *Semantic Interpreter*: Our SO driven semantic interpreter is a preliminary implementation that operates on JLink output. Informal tests show its interpretation to be accurate and its speed to be fast for text of reasonable complexity, such as for Navy Lessons Learned System documents.

4. *Feature extractor*: Not yet implemented
5. *Feature organizer*: Not yet implemented

6. *Case creation and indexing*: Our Document Extraction Workbench tool is useful for manually annotating the

primary components of a case from documents. It uses a handy drag and drop interface to provide XML markup of arbitrary complexity. However, we have not yet automated this process, or implemented case indexing tools.

5. Conclusion and Future Work

Documents such as manuals, logs, and reports are a primary source of cases for CBR applications. Case acquisition systems have predominantly focused on case indexing, but have ignored the important task of feature acquisition. In this paper, we introduced FACIT, a semi-automated knowledge extraction framework that uses deep NLP techniques to perform feature acquisition and relational feature organization *when the source documents are relatively unstructured and the ability to detect subtle nuances in language is crucial to extraction performance*. FACIT uses domain specific generative ontologies to create logical form representations for text documents of arbitrary complexity. We showed how these could be used to semi-automatically extract features and their relations *without* using a priori patterns.

In the future, we will develop, implement, and evaluate components for each of FACIT's steps, and complete our existing components. We will evaluate the accuracy of generating logical forms by processing a variety of source text, and investigate various machine learning techniques for extracting features from logical forms. Finally, we will formalize the algorithm for subsumption detection and assess the impact of implication rules for this task.

As FACIT is not yet fully implemented, we cannot present evidence for our claims. When completed, it could best be compared with other domain independent approaches that semi-automatically perform feature extraction, organization, and assignment from text documents. However, we are not aware of any other approach that addresses this complete set of problems.

Acknowledgements

Thanks to Rosina Weber, Karl Branting and our reviewers for excellent suggestions on an earlier version of this paper.

References

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications,* **7**, 39-59.

Aha, D.W., Breslow, L.A., & Munoz-Avila, H. (2001). Conversational CBR. *Applied Intelligence,* **14**(1), 9-32.

Ashley, K., & Lenz, M. (Eds.) (1998). *Textual case-based reasoning: Papers from the AAAI workshop* (Technical Report WS-98-12). Madison, WI: AAAI Press.

Baudin, C., & Waterman, S. (1998). From text to cases: Machine aided text categorization for capturing business reengineering cases. In (Ashley & Lenz, 1998).

Brill, E. (1995). Transformation-based tagger, V1.14. [http://www.cs.jhu.edu/~brill/RBT1_14.tar.Z]

Brüninghaus, S., & Ashley, K.D. (1999). Bootstrapping case base development with annotated case summaries. *Proceedings of the Third International Conference on Case-Based Reasoning* (pp. 59-73). Seeon, Germany: Springer.

Brüninghaus, S., & Ashley, K.D. (2001). The role of information extraction for textual CBR. *Proceedings of the Fourth International Conference on Case-Based Reasoning* (pp. 74-89). Vancouver (BC), Canada: Springer.

Burke, R.D., Hammond, K.J., Kulyukin, V., Lytinen, S.L., Tomuro, N., & Schoenberg, S. (1997). Question answering from frequently-asked questions files: Experiences with the FAQ Finder system. *AI Magazine,* **18**(1), 57-66.

Cowie, J., & Lehnert, W. (1996). Information extraction. *Communications of the ACM*, **39**(1), 80-91.

CSSI (1998). *Canon Multipass C-5500 User Manual*. Canon Computer Systems Inc.

Felbaum, C. (Ed.) (1998). *WordNet: An electronic lexical database*. Cambridge, MA: MIT Press.

Gupta K.M. (1997). Case base engineering for large-scale industrial applications. In B.R. Gaines & R. Uthurusamy (Eds.), *Artificial Intelligence in Knowledge Management: Papers from the AAAI Spring Symposium* (Technical Report SS-97-01). Stanford, CA: AAAI Press.

Gupta, K.M. (2001). Taxonomic conversational case-based reasoning. *Proceedings of the Fourth International Conference on CBR* (pp. 219-233). Vancouver (BC), Canada: Springer.

Gupta, K.M., & Aha, D.W. (2003). Nominal concept representation in sublanguage ontologies. *Proceedings of the Second International Workshop on Generative Approaches to the Lexicon* (Technical Report). Geneva, Switzerland: University of Geneva, School of Translation and Interpretation.

Gupta, K.M., Aha, D.W., & Moore, P. (2004). Automatically organizing indexing taxonomies from acquired features. Manuscript submitted for publication.

Gupta, K.M., Aha, D.W., & Sandhu, N. (2002). Exploiting taxonomic and causal relations in conversational case retrieval. *Proceedings of the Sixth European Conference on Case-Based Reasoning* (pp. 133-147). Aberdeen, Scotland: Springer.

Knight, K., & Luk, S. (1994). Building a large knowledge base for machine translation. *Proceedings of the American Association of Artificial Intelligence* (pp. 773-778). Seattle, WA: AAAI Press.

Lenz, M., Hübner, A., & Kunze, M. (1998). Textual CBR. In M. Lenz, B. Bartsch-Spörl, H.-D. Burkhard, & S. Wess (Eds.) *CBR technology: From foundations to applications.* Berlin: Springer.

Link (2003). The link parser application program interface (API). [http://www.link.cs.cmu.edu/link/api]

Minor, M, & Hübner, A. (1999). Semi-automatic knowledge acquisition for textual CBR. In R. Feldman & H. Hirsh (Eds.), *Text mining: Foundations, techniques, and applications: Papers from the IJCAI-99 Workshop*. Unpublished manuscript.

Pustejovsky, J. (1995). *The generative lexicon.* Cambridge, MA: MIT Press.

Russell, S., & Norvig, P. (1995). *Artificial Intelligence: A modern approach*. Englewood Cliffs, NJ: Prentice Hall.

Weber, R., Martins, A., & Barcia, R.M. (1998). On legal texts and cases. In (Ashley & Lenz, 1998).

Yang, C., Orchard, R., Farley, B., & Zaluski, M. (2003). Automated case base creation and management. *Proceedings of the Sixteenth International Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems*. Loughborough, UK: Springer.

Zavrel, J. & Daelemans, W., (1999). *Recent advances in memory-based part-of-speech tagging* (Technical Report 9903). Tilburg, Netherlands: Tilburg University, Faculty of Arts, Computational Linguistics and AI, Induction of Linguistic Knowledge Group.