The Role of Information Extraction for Textual CBR [*]

Stefanie Brüninghaus and Kevin D. Ashley

Learning Research and Development Center,
Intelligent Systems Program, and School of Law
University of Pittsburgh
3939 O'Hara Street; Pittsburgh, PA 15260 (USA)
steffi,ashley@pitt.edu

Abstract. The benefits of CBR methods in domains where cases are text depend on the underlying text representation. Today, most TCBR approaches are limited to the degree that they are based on efficient, but weak IR methods. These do not allow for reasoning about the similarities between cases, which is mandatory for many CBR tasks beyond text retrieval, including adaptation or argumentation. In order to carry out more advanced CBR that compares complex cases in terms of abstract indexes, NLP methods are required to derive a better case representation. This paper discusses how state-of-the-art NLP/IE methods might be used for automatically extracting relevant factual information, preserving information captured in text structure and ascertaining negation. It also presents our ongoing research on automatically deriving abstract indexing concepts from legal case texts. We report progress toward integrating IE techniques and ML for generalizing from case texts to our CBR case representation.

1 Motivation

CBR systems are particularly promising in domains like the law, ethics, business, customer support, medicine or intelligent tutoring, where human experts accumulate experiences and reason with precedents. However, these domains are not considered "natural CBR domains" (Leake 1998, p. 5), because the cases are recorded as text, which is notoriously difficult to use. To date, reasoning with text cases either requires considerable case engineering efforts or remains restricted to basic text retrieval based on information retrieval (IR) methods. In order to overcome this case representation gap, natural language processing (NLP) techniques are necessary. For more advanced CBR, including reasoning about the similarities and differences of partially matched cases for adaptation or argumentation, meaningful features beyond single words have to be elicited from the case texts. In the past, the brittleness and inefficiency of NLP systems made them all but inapplicable for CBR. However, research on Information Extraction (IE) has made remarkable progress, yielding more efficient and robust systems.

In this paper, we discuss how a state-of-the-art IE system, AutoSlog, can be used to extract relevant information from text cases and to derive a better text representation. While IE will facilitate building TCBR systems, fact extraction alone is not sufficient. In particular for interpretive CBR or more recent research trends, like Intelligent Lessons Learned systems, the case representation has to abstract from the bare facts of the case to its relevant fact patterns or lessons. In our SMILE system, we work toward integrating IE and ML methods for automatically assigning abstract indexing concepts to text cases.

[*] We thank Ellen Riloff for making AutoSlog available to us. This work would not have been possible without her program AutoSlog, which inspired many of the ideas reported here.

2 The Need for NLP/IE in Textual CBR

To date, most of the work in TCBR relies on available, robust and easy to use IR methods, which require minimal customization and knowledge engineering; see the collection in (Ashley & Lenz 1998). Other systems are CaseAdvisor (Racine & Yang 1997) in a tech-support application, and more recently DRAMA (Leake & Wilson 1999). DRAMA is particularly interesting because it combines CBR and IR methods for hybrid cases.

IR methods are extremely weak compared to the reasoning typically carried out in CBR, which includes the assessment of similarity among cases and the adaptation of partially matched cases. The text representation in IR is based only on the presence and absence of words; word order and text structure are not taken into account; similarity between documents is calculated based on superficial word frequency statistics. TCBR applications need to overcome these limitations, by adopting techniques for better text representation and by developing methods for mapping cases into structures that support the comparison of cases.

One way to improve upon the so-called *bag-of-words* (BOW) representation in IR is to capture word meaning. With a better text representation, a TCBR system can compare cases with similar content, but different words. The FallQ (Lenz, Hübner, & Kunze 1998) and FAQ FINDER (Burke *et al.* 1997) projects have used the lexical semantics in WordNet (Fellbaum 1998) to better represent the similarity among words. Experimental results from both projects show better case retrieval. However, WordNet was not designed to capture semantics at the text level (Fellbaum 1998, p. 7) and does not always correspond to everyday word use. We explored WordNet for our legal documents and found that this often causes undesirable inferences when one tries to find "correct" or domain-specific relations between words. In our research (Brüninghaus & Ashley 1999), we used an online domain-specific thesaurus instead, which lead to better performance for automatic indexing. We are currently working on (1) deriving a semantic representation of the language used in trade secret law cases based on a more comprehensive printed legal thesaurus (Burton 1992), and (2) integrating it with NLP tools. Recent work (Weber *et al.* 2001) has successfully used a domain ontology for TCBR.

While the most natural way to improve reasoning with text cases would be NLP, there has not been much research in TCBR that uses NLP methods to go beyond a BOW. FAQ FINDER made some use of NLP technology (Burke *et al.* 1997). A simple syntactic analysis was used both for part-of-speech tagging and for determining the type of question submitted by a user. The output of the parser was only used to further the use of WordNet and to better match documents, but not to improve text representation. Similarly, FallQ employed NLP techniques in the form of a part-of-speech tagger embedded in the stemming program to deal with the complexities of German inflection.

Two approaches have addressed the mapping of text cases into a representation that supports the comparison of cases. SPIRE aims at facilitating the task of a human indexer; Prudentia uses template mining to find basic case facts for case retrieval.

SPIRE's (Daniels & Rissland 1997) goal is to help a human locate the text passages related to CBR case features within long case texts. It combines CBR and IR methods in a two-step process. SPIRE first uses a Hypo-style CBR program to find cases to seed the IR system and select relevant new documents. In the second phase, SPIRE highlights those sections within a document to be indexed that are most likely to contain the information related to the CBR case representation. Thus, it significantly reduces the case engineering effort required to manually map text onto a symbolic representation.

Prudentia (Weber 1998) supports jurisprudential research by providing an effective case-based retrieval engine over a database of automatically indexed legal cases. The system uses a variety of pattern matching and template mining techniques to extract information from Brazilian criminal law cases. The cases are well-suited for this approach because they are comparatively short, follow strict structural rules, and use a uniform language. Compared to other legal CBR systems (Branting 1999), the case representation is relatively simple in that it includes mostly factual information like names and dates, supplemented with some more keyword-like categories. Likewise, basic pattern matching techniques have been used in FallQ (Lenz, Hübner, & Kunze 1998) to extract easily ascertainable feature/value pairs, in particular prices and technical information.

As this survey indicates, in many TCBR applications, factual information from the cases is relevant. Typically, these facts are names or locations, but may also include parametric information, like temperature or price values. For semi-structured texts, like job announcements, where a small number of standard phrases is used, pattern matching or template mining techniques (Weber 1998) are appropriate to extract this information. However, for texts without regular patterns and structure, like technical reports, these methods will not work.

Whenever the relevant information is embedded in free-form narratives, the deeper syntactic analysis of NLP/IE systems is necessary. In the past, the manual effort and large corpora of annotated training data required to set up these systems made them impractical for CBR indexing. However, IE programs have become more efficient and only require a few examples to automatically generate a problem-specific knowledge base.

To illustrate the benefits of NLP techniques for TCBR, imagine a hypothetical scenario. A well-known German carmaker has introduced a new level of service, through which customers can contact the company's hotline from their PDA without being put on hold. In response to a brief text message, the system retrieves a relevant past case. Assume a customer provides the following description: "I have a problem with my new 180. The wipe-wash system does not work, and water is oozing out at the oil dipstick." This short text includes the respective car, the reported problem and the observed symptoms. (For this example, assume the problem was caused by a rodent that chewed on the windshield washer fluid line.) A BOW representation of this case would exclude stopwords and remove suffixes, resulting in (180 dipstick new oil ooze problem system water wash wipe work). This is more like requesting a clean-up after an oil spill than the actual problem. A more meaningful representation of this example, and how NLP/IE methods can be employed to extract relevant information, is illustrated in Figure 1. The information in the target representation can be derived with techniques discussed here. This example shows three major tasks for an NLP/IE system in TCBR:

1. Extracting names and factual information,
2. Preserving information from text structure,
3. Ascertaining negation.

A fourth task can be identified for domains where the case representation is based on more general facts patterns or lessons, rather than concrete facts and specific events;

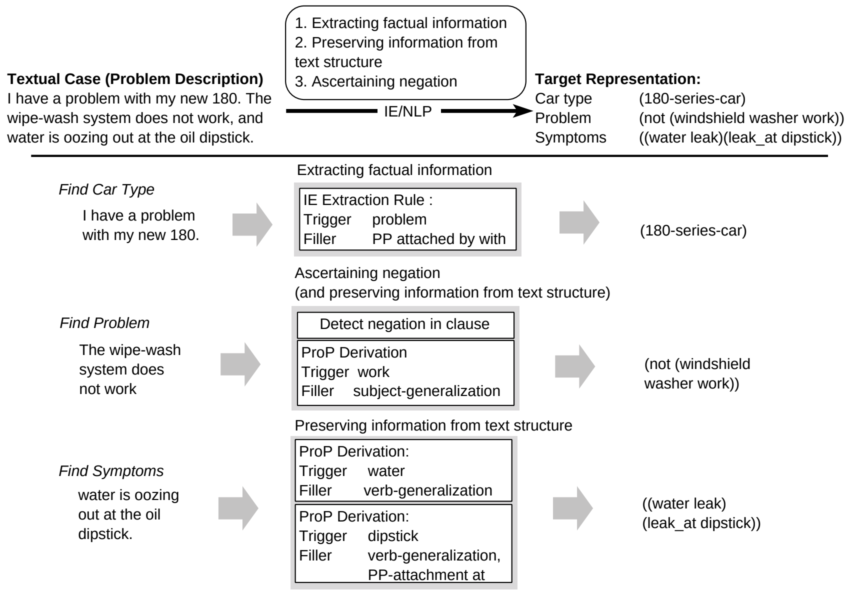

1. Extracting factual information
2. Preserving information from text structure
3. Ascertaining negation
Textual Case (Problem Description)
I have a problem with my new 180. The wipe-wash system does not work, and water is oozing out at the oil dipstick.
IE/NLP
Target Representation:
Car type (180-series-car)
Problem (not (windshield washer work))
Symptoms ((water leak)(leak_at dipstick))
Extracting factual information
Find Car Type
I have a problem with my new 180.
IE Extraction Rule :
Trigger problem
Filler PP attached by with
(180-series-car)
Ascertaining negation
(and preserving information from text structure)
Find Problem
The wipe-wash system does not work
Detect negation in clause
ProP Derivation
Trigger work
Filler subject-generalization
(not (windshield washer work))
Preserving information from text structure
Find Symptoms
water is oozing out at the oil dipstick.
ProP Derivation:
Trigger water
Filler verb-generalization
ProP Derivation:
Trigger dipstick
Filler verb-generalization, PP-attachment at
((water leak) (leak_at dipstick))

Fig. 1. Customer care example

here, an additional abstraction step using inductive learning techniques is required to derive the case representation.

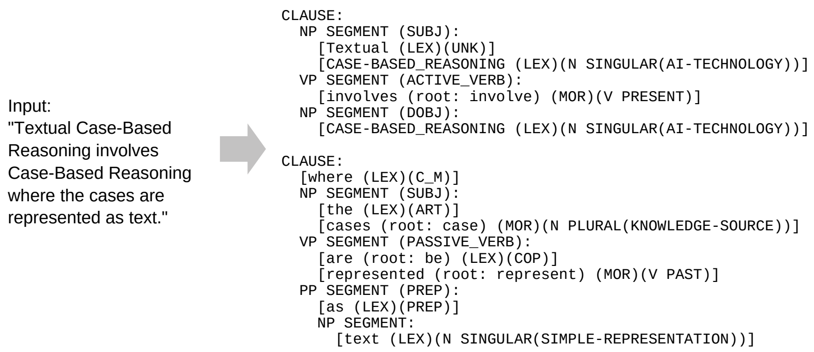
We will discuss the use of NLP/IE for the three major tasks in the next section, before investigating strategies for combining IE-based techniques and machine learning (ML) methods to induce criteria for finding more general concepts in text cases in Section 4.

3 Using AutoSlog for Case Indexing

In our research, we use a state-of-the-art NLP/IE system, AutoSlog (Riloff 1996), developed by Ellen Riloff at the University of Utah. AutoSlog employs a powerful heuristic sentence segmenter, Sundance, and a module for generating extraction rules from raw text and annotated examples. The remainder of this paper focuses on the use of AutoSlog. Presumably, our techniques are not limited to this particular program. Since most IE programs are not publicly available, however, we have not compared AutoSlog with other systems. http://www.isi.edu/∼muslea/RISE/ contains pointers to more IE-related information, including software and data collections.

Before we focus on the use of IE methods for TCBR, a brief introduction to the parsing in AutoSlog is in order. IE systems typically use partial parsing to find relevant syntactic constructs, which tends to be more robust and efficient than full parsing. A conventional parser performs a complete syntactic analysis to generate a full parse tree for a sentence, while a partial parser splits the sentence into clauses and only identifies local structures that can be determined reliably. Partial parsing thus helps overcome two major problems that hampered the use of NLP techniques in the past, their brittleness and the prohibitive time required to process realistic amounts of data.

Furthermore, in domains where the language is very complex, like in the law, partial parsing can help reduce errors. In the full parse of a long sentence, early errors get promoted and spread over the entire parse. Errors in a partial parse are more local, and the parser can recover from an error in the beginning of the sentence.



Input:
"Textual Case-Based
Reasoning involves
Case-Based Reasoning
where the cases are
represented as text."

```
CLAUSE:
  NP SEGMENT (SUBJ):
    [Textual (LEX)(UNK)]
    [CASE-BASED_REASONING (LEX)(N SINGULAR(AI-TECHNOLOGY))]
  VP SEGMENT (ACTIVE_VERB):
    [involves (root: involve) (MOR)(V PRESENT)]
  NP SEGMENT (DOBJ):
    [CASE-BASED_REASONING (LEX)(N SINGULAR(AI-TECHNOLOGY))]

CLAUSE:
  [where (LEX)(C_M)]
  NP SEGMENT (SUBJ):
    [the (LEX)(ART)]
    [cases (root: case) (MOR)(N PLURAL(KNOWLEDGE-SOURCE))]
  VP SEGMENT (PASSIVE_VERB):
    [are (root: be) (LEX)(COP)]
    [represented (root: represent) (MOR)(V PAST)]
  PP SEGMENT (PREP):
    [as (LEX)(PREP)]
    NP SEGMENT:
      [text (LEX)(N SINGULAR(SIMPLE-REPRESENTATION))]
```

Fig. 2. Output of Sundance parser

Figure 2 shows the output of Sundance, the parser in AutoSlog. It breaks complex sentences into clauses, and within each of these determines the constituents, including subject, verb, direct object and prepositional phrases. Sundance also identifies established terms ("case-based_reasoning"), noun phrases ("textual case-based_reasoning"), and retrieves semantic features ("ai-technology" and "simple-representation") associated with the words in its lexicon.

3.1 Extracting Factual Information with AutoSlog

AutoSlog can help extract names and factual information related to the case representation from text. In the example in Figure 1, the first piece of information to be extracted is the kind of car; the same problem may have different answers if it occurs with an SUV or a roadster. Here, the phrase "problem with" helps identify the car. Extracting factual information in this situation is a typical IE task. Like other IE systems, AutoSlog uses a rule-like mechanism. To better illustrate how this works, we will focus on a somewhat more difficult example from a different domain. Consider a typical sentence from our TCBR application, trade secret law: "Forcier developed an ink-processing technology," which is based on the *Forcier v. Microsoft* case in Figure 6. This sentence contains important information, the subject matter of the trade secret case. A human will identify the "ink-processing technology"; a TCBR system should do the same. AutoSlog can extract the trade secret with a rule "If the verb is developed, then extract the object."

AutoSlog has extraction rules[2], which consist of a *trigger* condition and the part of the sentence to be extracted as *filler*. The trigger is a word (or a combination of words)[3], and the filler is a constituent of the sentence, like those in the parse in Figure 2. In the example rule above, the trigger is the word "developed", and the filler is the direct ob-

[2] To prevent confusion with the terminology in Hypo (Ashley 1990), we prefer the generic phrase "extraction rule" over the Riloff's term, "caseframe".

[3] The trigger also includes the part-of-speech of the word, and in the case of verbs the form. For this discussion, we only focus on the presence of the word. We assume that part-of-speech can be ignored and that all verbs are active voice; passive voice will be indicated explicitly.

ject of the sentence. Figure 3 shows how AutoSlog first parses the input sentence, then checks whether rules are triggered, and returns the respective filler phrases.

A simple pattern-matching rule such as "Extract the nouns following the word 'developed' " may appear to work just as well, but is not an appropriate solution. If the sentence is more complicated, for instance "Forcier developed after many years of research an ink-processing technology," AutoSlog will still return the correct filler, whereas the pattern-matching rule would fail.

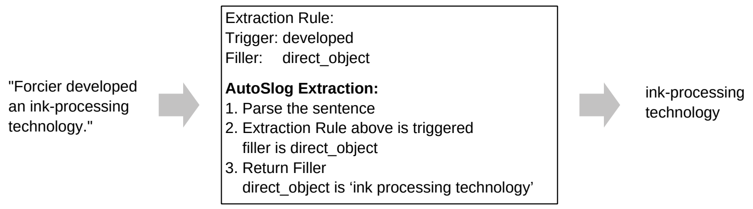

"Forcier developed an ink-processing technology."
Extraction Rule:
Trigger: developed
Filler: direct_object
AutoSlog Extraction:
1. Parse the sentence
2. Extraction Rule above is triggered
filler is direct_object
3. Return Filler
direct_object is 'ink processing technology'
ink-processing technology

Fig. 3. Extracting information with AutoSlog

Crafting extraction rules by hand is usually a very time-consuming task, and for most applications would be prohibitively expensive. Clearly, a method for automating this process is mandatory for the practical use of an IE system. Given an example sentence and the target information, AutoSlog can reverse the extraction process and derive extraction rules automatically. It first identifies the part of the sentence that contains the target information, which will become the "then extract" filler part of the extraction rule. It then uses a set of heuristics to determine the appropriate trigger condition, preferably the verb of the sentence. Figure 4 illustrates how the extraction rule used above can be derived from the sentence "Houser developed a nut-spinner" (from *Houser v. Snap-on Tools*), where the filler was the secret invention, the "nut-spinner". For more detail on the automatic generation of extraction rules; see (Riloff 1996).

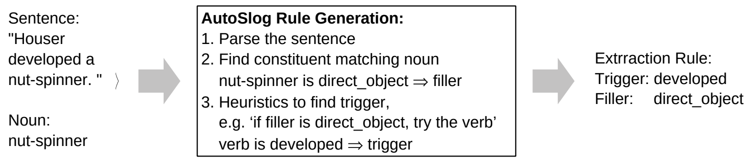

Sentence:
"Houser developed a nut-spinner. "
Noun:
nut-spinner
AutoSlog Rule Generation:
1. Parse the sentence
2. Find constituent matching noun
nut-spinner is direct_object ⇒ filler
3. Heuristics to find trigger,
e.g. 'if filler is direct_object, try the verb'
verb is developed ⇒ trigger
Extrraction Rule:
Trigger: developed
Filler: direct_object

Fig. 4. Deriving an Extraction Rule from an example

This function of AutoSlog can significantly facilitate the development of a CBR system. Even with a small number of training data, AutoSlog can produce a fairly good set of extraction rules. It should be pointed out that with real-world texts as input, the rules are sometimes too general; see for instance the second and fourth example in Table 1. Statistical methods can be used to filter out those rules; see (Riloff 1996).

While there are other systems that can derive extraction rules from examples in a somewhat similar fashion, AutoSlog is unique in that it can generate extraction rules even if no target filler is given. If only a sentence is given as input to the rule generation module, the system parses the sentence and collects all noun phrases as candidate information to be extracted. It then generates the applicable extraction rules for every noun phrase. This function is helpful for the generation of the Propositional Patterns introduced in Section 3.2.

Training Sentence	Product	Extraction Rule		
		Nr.	Trigger	Filler
SDRC began developing a computer program called NIESA.	computer program	(1)	developing	d_object
	NIESA	(2)	called	d_object
The plaintiff manufactures adhesive tape, including a “masking tape”	adhesive tape	(3)	manufactures	d_object
	masking tape	(4)	including	d_object

Table 1. Examples of extraction rules generated from squibs

The extraction of factual information with AutoSlog can be useful in a wide variety of CBR applications, wherever features beyond single words from a lexicon are relevant for the case-based reasoner. In a business decision-support application, information about prices and price changes can be extracted from AP news articles. In medical applications, IE methods can be used to extract symptoms or diagnoses from clinical records (Sonderland *et al.* 1995). For instance, from the sentence “The patient continues to have a high, uncontrollable fever and severe chills,” after (Sonderland *et al.* 1995), the symptom “high, uncontrollable fever” would be extracted. Another application where IE could be applied is SPIRE. One of the target features in its bankruptcy domain is the profession of the debtor. An example sentence from (Daniels 1997) is “Debtor had [...] secured employment as a receptionist-secretary,” where the profession is “receptionist-secretary”. From this, an extraction rule that extracts the prepositional phrase triggered by “employment as” can be derived.

AutoSlog’s extraction rules can work quite well for identifying information for a CBR application even without any manual intervention; see Section 4 for a preliminary experiment. However, they are not a magical solution. The relevant information has to be embedded in a known and recognizable context. Also, AutoSlog’s extraction rules tend to be overly general. They will be more accurate where the target objects play distinct roles in the domain. Parser errors, which are fairly common even with Sundance, can cause erroneous extraction rules, too. Generally, the accuracy of AutoSlog will decrease for ungrammatical or badly written text.

In Section 4, we will go beyond extracting names and show how these IE methods can be used in a novel way to make cases more useful by generalizing from individual names and entities to roles for the indexing of legal cases.

3.2 Preserving Information from Text Structure in ProPs

For CBR cases, it is usually necessary to represent what happened in a case. The text representation, whether it is used in retrieval or for automatic indexing, has to preserve the meaning captured in the word order and text structure. IR-based methods discard all syntax-related information needed to find the relevant events, whereas IE methods can be adapted to capture patterns of actions as well as negation. Typically, IE systems have been designed to extract concrete names and objects to fill slots in templates, and not to find more abstract actions or patterns. However, we think AutoSlog can be adapted for extracting not only objects but also patterns related to actions. It may be possible to accomplish this even without the need for a human to craft elaborate rules by hand.

We hope to achieve this by combining the trigger and a generalization of the filler of an extraction rule into single, more expressive features, which we call *Propositional*

Patterns (ProPs). This representation is intended to capture information such as "who did what?" or "what was it done to?" ProPs are created on the fly, and only capture fairly shallow syntactic relations among words, in contrast to a representation of entire events in scripts or frames (Jurafsky & Martin 2000). The ProPs are flexible, because they focus only partial views of a sentence. Furthermore, they generalize from the particular words to more general concepts, which makes them more useful for comparing different textual cases.

In the luxury car example, the car owner's observations are summarized in the clause "Water is oozing out at the oil dipstick." In this example, it is clearly important that water, and not oil, was spotted, and where the liquid was observed. Finally, it is desirable to generalize from "ooze" to a more technical term (we assume here leak is a generalization of ooze). The resulting ProPs and the rules to derive them are shown in Figure 1. Under "Find Symptoms", the ProPs are (water leak) and (leak_at dipstick).

ProPs may be useful for a variety of TCBR applications because they are more expressive features than single words, and at the same time support generalizing from the particular wording of the case text.

Recorded lessons learned episodes, for instance, typically contain an event description and a summary of the intended lesson. A lesson could be "The incident status should be displayed by the Incident Command System." (after US Coast Guard CG-SAILS collection). From the BOW representation (command display incident status system), it is not clear what lesson can possibly be learned. Conceivably, ProPs may better capture who should make the information available as (Incident_Command_System publicize) and what measure should be taken as (publicize incident_status). (Weber *et al.* 2001) contains an example for the condition when a lesson can be reused, "A civilian airport was transformed into an intensive military operation area." For this sentence to be meaningful, it is crucial to preserve what was transformed. One way to accomplish this could be the ProP representation (civilian_airport transformed_passive) and (transformed_passive_into military_operation_area).

Another possible application for ProPs is in SIROCCO (Ashley & McLaren 2001), which reasons with engineering ethics cases. The cases are represented in terms of manually extracted Fact Primitives. The sentence "Engineer A has a degree in mechanical engineering and has performed services almost exclusively in mechanical engineering." corresponds to "Engineer A ⟨specializes_in⟩ Mechanical engineering". The related ProPs would be (engineer_a practice) and (practice_in mechanical_engineering), assuming that the domain ontology generalizes from perform services to practice. While the ProPs are not as powerful as the Fact Primitives, they capture most of the relevant information and could eliminate the need to manually encode every single document.

Whether ProPs are applicable and will improve performance depends on the type of texts; certainly not all written materials are suitable for this approach. For a system like FallQ (Lenz, Hübner, & Kunze 1998), where the queries are very short and consist mostly of product identifiers, little improvement over IR-based methods can be expected. Generally, ProPs will be more useful to the degree that the texts follow the Elements of Style (Strunk & White 1979). Also, typical NLP problems, for instance pronoun resolution, will remain an obstacle and cannot be overcome by ProPs.

3.3 Ascertaining Negation with Sundance

Apart from extracting multi-word features, NLP can also be important for dealing with negation. In the prevalent IR methods, negation is disregarded. Stopwords, including "no" and "not", are deleted, even though experimental evidence shows that small words like these can make a big difference for determining what documents are about (Riloff 1995). This may be especially true in finding more factual evidence in text documents.

Negation is considered a hard NLP problem, in part because even when evidence for negation is detected, it can be difficult to determine what is negated. Consider the sentence from our luxury car example, "The wipe-wash system does not work, and water is oozing out at the oil dipstick." In this sentence, only the functionality of the windshield washer is negated, but not the observed symptoms. Relying only on the presence of "not" is not sufficient, one has to take the scope of the clause into account.

For example, a typical sentence, which with minor variations can be found in a number of legal cases in the Case Database of our CATO program (Aleven 1997), is "The information was unique and not generally known in the industry." This sentence is evidence for CATO's CBR indexing concept F15, Unique-Product, because the information was *not* generally known. If everything in the sentence remained identical, and only the word "not" were moved in front of "unique", the sentence would not be evidence for F15, but rather for the opposite concept F20, Info-Known-to-Competitors. Simply relying on the presence of words like "not" in a sentence is insufficient for legal case texts.

We have found that negation usually covers a clause. In the example sentence above, the meaning changed when the word "not" was moved from one clause to the other. Based on this assumption, negation can be ascertained in a two-step process. First, Sundance can be used to break up a complex sentence into clauses, as in Figure 2. Second, if negation is found in any part of a clause, it can be assumed to cover all ProPs derived from that clause.

Methods for correctly representing negation are relevant for many TCBR applications. They are particularly important for the emerging area of medical CBR applications (Bellazzi *et al.* 1999), where patient records explicitly mention the absence of symptoms.

In customer service applications, it is important to correctly extract negated facts from textual descriptions of the cases. For retrieving an accurate response in the printer trouble-shooting situation discussed in (Aha, Maney, & Breslow 1998), it is crucial to know whether the user observes black streaks on the printout or not.

Likewise, finding negation can be relevant for automatically adding cases to case-based collaborative filtering applications (Burke 2000). Entree's cases are restaurant descriptions, which could be gleaned from online reviews. Negation will be needed to capture, for instance, evidence related to noise in sentences like "It is not a noisy restaurant," and "LuLus can get extremely noisy, [. . .]" (from the Pittsburgh Post-Gazette).

The fairly straight-forward method suggested here for ascertaining negation does not solve all problems related to negation, like double negation or expressions of doubt. One can easily come up with a list of exceptions for different domains. However, our goal is not to cover all possible forms of negation; instead, we try to find those cases where we can be reasonably certain and represent them correctly. It seems most useful to have a weak, but reliable method, which can be applied in all domains and expanded upon as necessary. We will assess empirically whether this approach is sufficient for our task.

4 Application: Finding Factors for CATO with SMILE

For many CBR applications, the relevant features of a case are not concrete facts, like the names, locations or numbers typically extracted by an IE system, and the CBR task is more complex than retrieving text cases. Instead, these CBR systems reason with more abstract fact patterns or relevant lessons contained in a case.

One such system is CATO (Aleven 1997), an intelligent tutoring environment for teaching skills of making arguments with cases to beginning law students. In the core of CATO is an expert model of case-based argumentation. The system has a collection of about 150 cases from the domain of trade secret misappropriation. These cases are represented in terms of 26 binary Factors, prototypical, abstract fact patterns that tend to strengthen or weaken the plaintiff's claim. For each case, we have a case summary, called a squib, and the court's full-text opinion. CATO can generate arguments comparing and contrasting cases in terms of Factors. In addition, it uses knowledge about higher-level legal knowledge to reason about abstract issues. A major obstacle for the practical use of CATO is the prohibitive cost of developing and maintaining a large case-base.

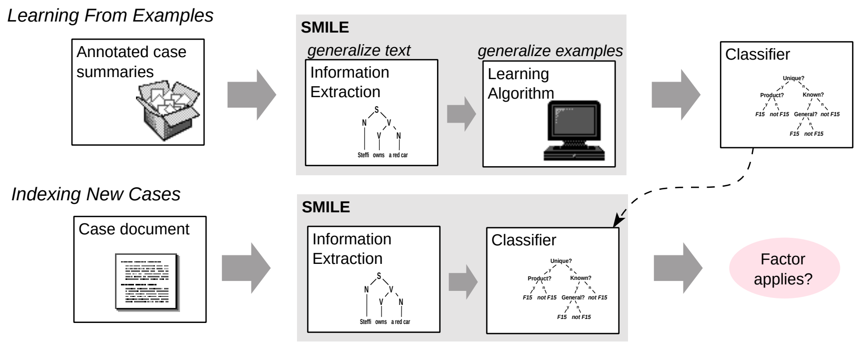


Fig. 5. Overview of SMILE

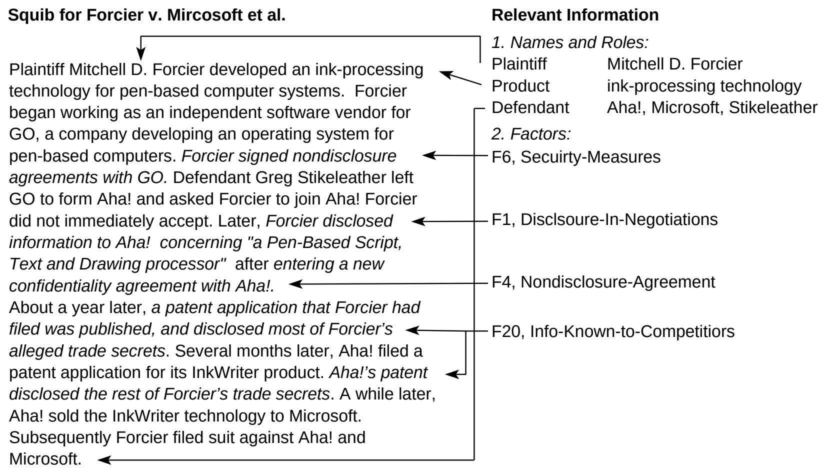
We address this problem with SMILE[4], a program to automatically index new cases. Figure 5 shows the design of SMILE. In the *Learning from Examples* phase, SMILE uses a collection of manually marked-up squibs as examples of how indexing should be done and learns classifiers, which will be used in the *Indexing New Cases* phase to classify new case texts under CATO's Factors. We use an inductive learning approach, because the prevalent statistical text learning algorithms, which were designed to work with massive amounts of data and rather simple concepts, are not applicable here (Brüninghaus & Ashley 1999; 1997). In a CBR context, manually annotated training data is scarce, while the indexing concepts are relatively complex. Compared to other text classification applications, CATO's Case Database of 150 documents is a very small collection, and the Factors are hard concepts to learn.

One obstacle is that a BOW representation does not allow the classifier to generalize from the examples to the abstract fact patterns related to the Factors. We therefore integrate the IE techniques in AutoSlog as a first step to generalize from the text to a more

[4] SMart Index LEarner

suitable example representation that captures the information relevant for finding Factors. The texts are generalized in two steps, using the techniques for better representing text cases introduced in Section 3. First, we substitute names with roles. Part of this step has been implemented, and we present a preliminary evaluation below. Second, we are planning to capture patterns of actions in ProPs and ascertain negation.

The input to SMILE are squibs, in which the sentences that provide evidence for the Factors are marked up. An example is the *Forcier* squib in Figure 6, which summarizes the facts from a real case, *Mitchell F. Forcier v. Microsoft Corporation, et. al*, 123 F.Supp2d 520. This squib reflects the relevant wording of the original court opinion.



Squib for Forcier v. Mircosoft et al.

Plaintiff Mitchell D. Forcier developed an ink-processing technology for pen-based computer systems. Forcier began working as an independent software vendor for GO, a company developing an operating system for pen-based computers. *Forcier signed nondisclosure agreements with GO.* Defendant Greg Stikeleather left GO to form Aha! and asked Forcier to join Aha! Forcier did not immediately accept. Later, *Forcier disclosed information to Aha! concerning "a Pen-Based Script, Text and Drawing processor"* after *entering a new confidentiality agreement with Aha!.* About a year later, *a patent application that Forcier had filed was published, and disclosed most of Forcier's alleged trade secrets*. Several months later, Aha! filed a patent application for its InkWriter product. *Aha!'s patent disclosed the rest of Forcier's trade secrets*. A while later, Aha! sold the InkWriter technology to Microsoft. Subsequently Forcier filed suit against Aha! and Microsoft.

Relevant Information

1. Names and Roles:

Plaintiff	Mitchell D. Forcier
Product	ink-processing technology
Defendant	Aha!, Microsoft, Stikeleather

2. Factors:

F6, Secuirty-Measures

F1, Disclsoure-In-Negotiations

F4, Nondisclosure-Agreement

F20, Info-Known-to-Competitiors

Fig. 6. Example squib for the *Forcier v. Microsoft* case

4.1 Generalizing from Names and Objects to Roles

For comparing cases in CATO, the particular names of the parties are irrelevant features. More important is the role that a named party plays in the case. Similarly, product names and details only make similar objects appear to be different. Consider the following three sentences. They only have the word "to" in common, and there is no obvious pattern.

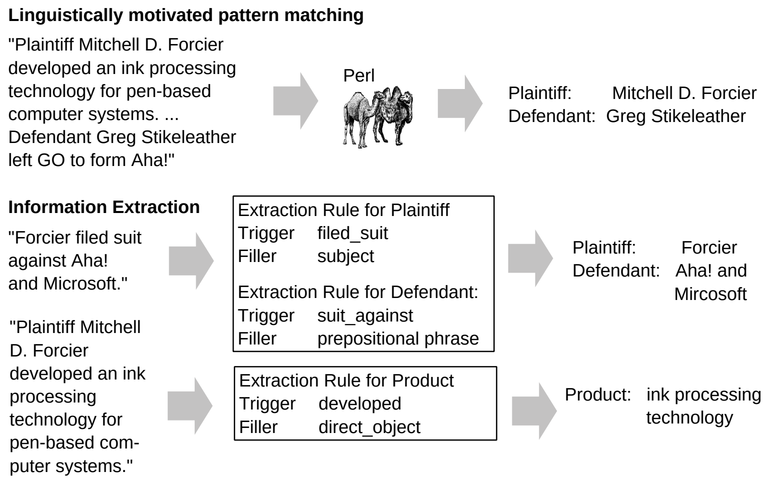
– Forcier disclosed information to Aha! (from Figure 6)
– Revcor gave its complete drawings to Marchal and Fulton.
– Hisel sent a letter to Chrysler explaining his idea for a glass-covered holder for license plates.

If we know more about the original cases, however, the three sentences are instances of a pattern. We can generalize them by replacing the product-related information and substituting the names by their role in the lawsuit. (I.e., Forcier, Revcor, Hisel become Plaintiff; Aha!, Marchal and Fulton, and Chrysler become Defendant.)

– Plaintiff disclosed the information to defendant.
– Plaintiff gave the information to defendants.
– Plaintiff sent a letter to defendant explaining the information.

In the modified sentences, we can find the common pattern that (1) plaintiff disseminated something, that (2) product-related information was disseminated, and that (3) the information was given to defendant. Each of these sentences is evidence for the applicability of CATO's Factor F1, Disclosure-In-Negotiations.

In SMILE, we use domain-specific heuristics implemented in Perl that exploit the typical wording and linguistic constructs like appositions to identify the plaintiff and defendant in a case; see Figure 7. This can be supplemented with AutoSlog's Extraction Rules. We have not yet conducted a formal evaluation.



Linguistically motivated pattern matching
"Plaintiff Mitchell D. Forcier developed an ink processing technology for pen-based computer systems. ... Defendant Greg Stikeleather left GO to form Aha!"
Perl
Plaintiff: Mitchell D. Forcier
Defendant: Greg Stikeleather
Information Extraction
"Forcier filed suit against Aha! and Microsoft."
Extraction Rule for Plaintiff
Trigger filed_suit
Filler subject
Extraction Rule for Defendant:
Trigger suit_against
Filler prepositional phrase
Plaintiff: Forcier
Defendant: Aha! and Mircosoft
"Plaintiff Mitchell D. Forcier developed an ink processing technology for pen-based computer systems."
Extraction Rule for Product
Trigger developed
Filler direct_object
Product: ink processing technology

Fig. 7. Extracting the names of plaintiff and defendants and the product from the *Forcier* squib

We then substitute the extracted names and object references with their roles in the lawsuit, thereby generalizing from individual instances towards the goal concepts. This measure also adds information from the overall case context to the sentences.

The example sentences show that generalizing from the individual products and inventions to their more general role for the case can benefit finding Factors. Otherwise, product specific identifiers would prevent the comparison of different cases. Product-related information cannot be extracted with simple pattern matching techniques. As argued above, IE techniques will be necessary.

In a preliminary experiment, we tried to find out whether AutoSlog's extraction rules for product-related information can be derived from CATO's squibs without manual intervention and fine-tuning, like filtering out overly general rules or correcting mistakes by Sundance. For the experiment, we manually extracted product names and identifiers from CATO's squibs. We then split the collection into two subsets, and in turns derived extraction rules from one set of examples, which included the squibs and the manually extracted product information as training data. We tested these rules on the other set.

Table 1 has some examples, which suggest that some extraction rules are very accurate, while others are overly general. Extracting the direct object of the verb "including" yields many product names, but also returns a large number of references to persons. We also found some incorrect extraction rules, which were apparently the result of a parser error. One rule extracted the direct object when triggered by the word "signed". In our experiment, we therefore filtered out all extracted fillers that were automatically recognized by Sundance as persons, contract terms or dates.

In scoring the experiment, we did not consider it an error when only part of a complex noun phrase or a general term were extracted. We calculated recall as the portion of all manually marked-up names and products that were found, and precision as the portion of extracted fillers that contained the marked-up names, or other product-related information. The results, macroaveraged over each squib, were precision 66.15% ($\sigma = 0.12$) and recall 64.82% ($\sigma = 0.05$). Notice that only 75 cases, half of the collection, were used to derive the extraction rules for the test set, and that we did not do any manual fine-tuning of the extraction rules. Such adjustments are fairly common for IE systems. Increasing the number of training instances conceivably would lead to higher recall, while removing overly general or incorrect extraction rules is likely to increase precision.

4.2 Using ProPs To Classify Case Texts

Once names are substituted by roles, the next step is to ascertain negation and represent information captured in the text structure. The discussion here will focus on ProPs; for an example how negation is relevant for finding Factors, see Section 3.3.

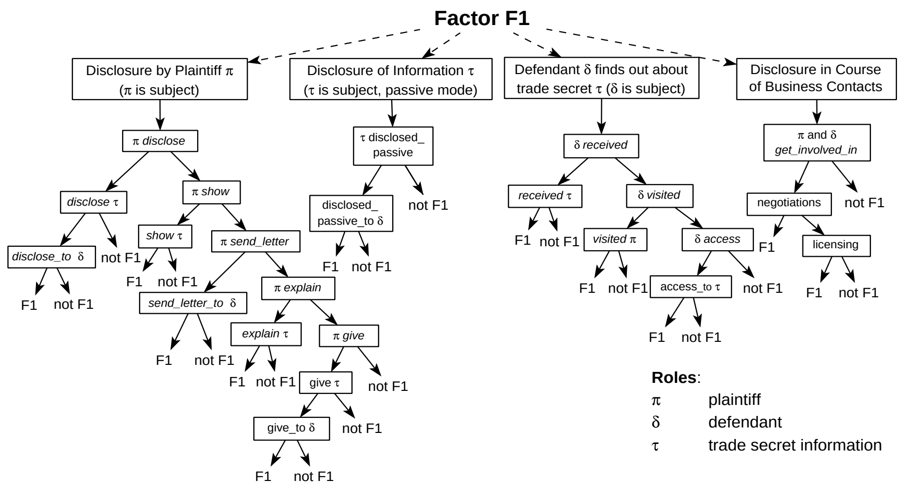


Fig. 8. Manually generated classification tree for Factor F1, Disclosure-in-Negotiations; arrow to the left indicates the feature is present, arrow to the right indicates the feature is absent.

As mentioned above, in the *Forcier* squib, the sentence “Plaintiff disclosed information to defendant . . .” contains evidence for the Factor F1, Disclosure-In-Negotiations. Another sentence from this squib is “Defendant’s patent disclosed most of plaintiff’s information.” It shows that Factor F20, Info-Known-to-Competitors, applies, but is not related to Factor F1. The relevant difference with respect to Factor F1 between these sentences is not the word “patent,” but who disclosed the information and to whom. For Factor F1 to apply, the information has to be disclosed by plaintiff to defendant. If the sentences are represented by ProPs, as (plaintiff disclosed) (disclosed information) (disclosed_to defendant) and (patent disclosed) (disclosed information), respectively, they can be readily distinguished based on who disclosed the information.

We are currently working on how to derive ProPs automatically. To give an intuition as to the goal of this research, we manually generated the type of classifier we hope to learn with SMILE. We first collected the sentences that contain evidence for Factor F1 from CATO's squibs and replaced all names by roles. Then, we abstracted from the actions, which allowed us to derive ProPs. From this representation, we generated the classification tree in Figure 8. Since this was done by hand, we focussed on the positive examples only and did not use a learning algorithm; instead, we relied on a common-sense covering strategy. The sentence "Forcier disclosed the information to Aha!", represented as (plaintiff disclosed) (disclosed information) (disclosed_to defendant), would be classified correctly as an instance of F1 through the leftmost branch in this tree. Without generalizing from the text by manually emulating the techniques introduced in Section 3, it would be impossible to derive this classification tree for F1.

5 Summary

In this paper, we discussed the limitations of IR-based TCBR approaches and how they can be addressed with a state-of-the-art NLP/IE tool, AutoSlog. Specifically, we identified three tasks where NLP/IE can improve TCBR systems. They are (1) extracting names and factual information, (2) preserving information captured in the syntax of the documents, and (3) ascertaining negation. In particular, we discussed how a new type of feature, called Propositional Patterns, can be derived from syntactic relations among words to capture information like "who did what?" These tasks were motivated with an example. We also drew examples from a variety of TCBR systems to demonstrate how these techniques can help overcome the limitations of a BOW representation.

After presenting the general techniques, we focussed on our own TCBR program SMILE for identifying CATO's Factors in legal cases. The Factors are abstract fact patterns, used to compare cases and reason about the differences between partially matched cases. We are working on integrating the three IE-based techniques above and ML methods for better generalizing from the legal case opinions. Our approach in SMILE consists of two steps. First, SMILE generalizes from the text using NLP/IE methods. We replace the names of the parties and product-related information by their role in the lawsuit, derive ProPs to preserve some of the relevant information in the text, and ascertain negation. Second, SMILE uses an ML algorithm to further generalize from examples to more abstract fact patterns in a classifier.

References

Aha, D.; Maney, T.; and Breslow, L. A. 1998. Supporting dialogue inferencing in conversational case-based reasoning. In *Proceedings of the Fourth European Workshop on Case-Based Reasoning*. Springer. LNAI 1488.

Aleven, V. 1997. *Teaching Case-Based Argumentation through a Model and Examples*. Ph.D. Dissertation, University of Pittsburgh.

Ashley, K., and Lenz, M., eds. 1998. *Textual Case-Based Reasoning, Papers from the AAAI-98 Workshop*. AAAI Press. Technical Report WS-98-12.

Ashley, K., and McLaren, B. 2001. Helping a CBR Program Know What it Knows. In *Proceedings of the Fourth International Conference on Case-Based Reasoning*. Springer.

Ashley, K. 1990. *Modeling Legal Argument, Reasoning with Cases and Hypotheticals*. MIT-Press.

Bellazzi, R.; Montani, S.; Portinale, L.; and Riva, A. 1999. Integrating Case-Based and Rule-Based Decision Making in Diabetic Patient Management. In *Proceedings of the Third International Conference on Case-Based Reasoning*. Springer. LNAI 1650.

Branting, L. 1999. *Reasoning with Rules and Precedents - A Computational Model of Legal Analysis*. Kluwer Academic Publishers.

Brüninghaus, S., and Ashley, K. 1997. Using Machine Learning for Assigning Indices to Textual Cases. In *Proceedings of the Second International Conference on Case-Based Reasoning*. Springer. LNAI 1266.

Brüninghaus, S., and Ashley, K. 1999. Bootstrapping Case Base Development with Annotated Case Summmaries. In *Proceedings of the Third International Conference on Case-Based Reasoning*. Springer. LNAI 1650.

Burke, R.; Hammond, K.; Kulyukin, V.; Lytinen, S.; Tomuro, N.; and Schonberg, S. 1997. Question-Answering from Frequently-Asked Question Files: Experiences with the FAQ-Finder System. *AI Magazine* 18(1):57–66.

Burke, R. 2000. A Case-Based Approach to Collaborative Filtering. In *Proceedings of the Fifth European Workshop on Case-Based Reasoning*. Springer. LNAI 1898.

Burton, W. 1992. *Legal Thesaurus*. Simon & Schuster Macmillan.

Daniels, J., and Rissland, E. 1997. Finding Legally Relevant Passages in Case Opinions. In *Proceedings of the Sixth International Conference on AI and Law*. ACM Press.

Daniels, J. 1997. *Retrieval of Passages for Information Reduction.* Ph.D. Dissertation, University of Massachusetts, Amherst, MA.

Fellbaum, C., ed. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Jurafsky, D., and Martin, J. 2000. *Speech and Language Processing*. Prentice Hall.

Leake, D., and Wilson, D. 1999. Combining CBR with Interactive Knowledge Acquisition, Manipulation and Reuse. In *Proceedings of the Third International Conference on Case-Based Reasoning*. Springer. LNAI 1650.

Leake, D., ed. 1998. *Case-Based Reasoning: Experiences, Lessons & Future Directions*. MIT Press.

Lenz, M.; Hübner, A.; and Kunze, M. 1998. Textual CBR. In Lenz, M.; Bartsch, B.; Burkhard, H.-D.; and Wess, S., eds., *Case-Based Resaoning Technology*. Springer. LNAI 1400.

Racine, K., and Yang, Q. 1997. Maintaining Unstructured Case Bases. In *Proceedings of the Second International Conference on Case-Based Reasoning*. Springer. LNAI 1266.

Riloff, E. 1995. Little Words Can Make a Big Difference for Text Classification. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press.

Riloff, E. 1996. Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*. AAAI Press.

Sonderland, S.; Aronow, D.; Fisher, D.; Aseltine, J.; and Lehnert, W. 1995. Machine Learning of Text Analysis Rules for Clinical Records. Technical Report CIIR TC-39, University of Massachusetts, Amherst, MA.

Strunk, W., and White, E. 1979. *The Elements of Style*. Macmillan Publishing Co.

Weber, R.; Aha, D.; Sandhu, N.; and Munoz-Avila, H. 2001. A textual case-based reasoning framework for knowledge management applications. In *Proceedings of the Ninth German Workshop on Case-Based Reasoning*. Shaker Verlag.

Weber, R. 1998. *Intelligent Jurisprudence Research.* Ph.D. Dissertation, Federal University of Santa Catarina, Brazil.