Improving the Representation of Legal Case Texts with Information Extraction Methods

Stefanie Brüninghaus and Kevin D. Ashley
Learning Research and Development Center
Intelligent Systems Program and School of Law
University of Pittsburgh, Pittsburgh PA 15260
steffi@pitt.edu, ashley@pitt.edu

ABSTRACT

The prohibitive cost of assigning indices to textual cases is a major obstacle for the practical use of AI and Law systems supporting reasoning and arguing with cases. While progress has been made toward extracting certain facts from well-structured case texts or classifying case abstracts under Key Number concepts, these methods still do not suffice for the complexity of indexing concepts in CBR systems.

In this paper, we lay out how a better example representation may facilitate classification-based indexing. Our hypotheses are that (1) abstracting from the individual actors and events in cases, (2) capturing actions in multi-word features, and (3) recognizing negation, can lead to a better representation of legal case texts for automatic indexing. We discuss how to implement these techniques with state-of-the-art NLP tools. Preliminary experimental results suggest that a combination of domain-specific knowledge and information extraction techniques can be used to generalize from the examples and derive more powerful features.

1. INTRODUCTION

AI and Law research has produced numerous sophisticated models and formalizations for reasoning and arguing with cases (Ashley 1990; Aleven 1997; Rissland, Skalak, & Friedman 1996; Branting 1999); see also (Prakken & Sartor 1998). Despite their potential benefits, however, none of these approaches has lead to systems used in legal practice, for instance as intelligent assistants that help in exploring case law or in generating arguments with cases.

One of the major obstacles for the use of those models in law offices is the prohibitive cost of representing cases in a form that allows an AI program to reason with them. Up to now, case indexing and information extraction (IE) for case-based reasoning (CBR) has been a manual effort. Unless methods are developed that support creating and maintaining large casebases, these systems will remain restricted to the research world.

This problem can be addressed by automatically extracting features from textual cases. The Salomon (Moens 2000) and Prudentia (Weber 1999) projects demonstrated how relevant facts, like names, dates or references to statutes, can be gleaned from civil law criminal cases and used to perform some basic case retrieval. Where the cases are as well-structured as in these applications, key information can be easily spotted, using for instance location and cue-words. For many CBR domains, like trade secret law, however, case structure differs significantly across cases. In addition, the information extracted in both systems was more concrete and easier to find than the abstract indexing concepts typical for CBR systems.

Our SMILE[1] project explores new methods to extract information from and index textual cases. It uses a set of marked-up case summaries as examples in a machine learning approach to classify new cases under indexing concepts. In this paper, we illustrate how a better example representation may help improve this classification. We hypothesize that (1) abstracting from the individual actors and events of a case to generic roles, (2) capturing actions in multi-word features, and (3) determining the presence and scope of negation support a more powerful representation enabling classifiers to generalize accurately from training examples.

In designing SMILE we set out to combine state-of-the-art NLP tools and domain-specific knowledge, which has not been done in previous work. For instance, the goal of the SPIRE system (Daniels & Rissland 1997) also was to overcome the knowledge-engineering bottleneck of indexing cases for a HYPO-style CBR system. SPIRE successfully used a small collection of marked-up text segments to help direct a human indexer to relevant passages in long cases. For this, SPIRE employed the relevance feedback and passage retrieval mechanisms of an existing information retrieval (IR) system. It did not, however, integrate any additional background knowledge or linguistic information.

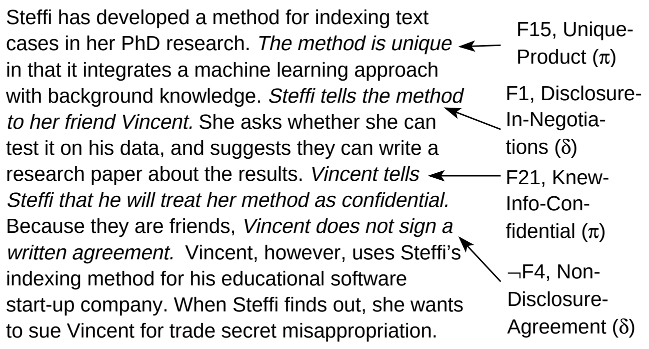
The rest of this paper is organized as follows. In Section 2, we show an example case and introduce our case-based argumentation program CATO. In Section 3, we present the learning approach for assigning indices in SMILE. In Section 4, we discuss why the three techniques for an improved case representation may lead to better indexing for legal cases. In Sections 5, we introduce the NLP tools we are using. In Section 6, we demonstrate our approach to using NLP methods and background knowledge to implement these measures, and present some preliminary results. In Section 7, we summarize the most important issues raised.

2. REASONING WITH FACTORS IN CATO

In order to illustrate our ideas with a single textual example, we present the hypothetical case of *Steffi v. Vincent* in Figure 1. This example is patterned after the *Mason* case, see (Aleven 1997). The text is comparable to the factual summaries, or squibs, that law

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICAIL-2001 St. Louis, Missouri USA
Copyright 2001 ACM 1-58113-368-5/01/0005 $5.00.

[1]SMart Index LEarner



Steffi has developed a method for indexing text cases in her PhD research. *The method is unique* in that it integrates a machine learning approach with background knowledge. *Steffi tells the method to her friend Vincent.* She asks whether she can test it on his data, and suggests they can write a research paper about the results. *Vincent tells Steffi that he will treat her method as confidential.* Because they are friends, *Vincent does not sign a written agreement.* Vincent, however, uses Steffi's indexing method for his educational software start-up company. When Steffi finds out, she wants to sue Vincent for trade secret misappropriation.

F15, Unique-Product (π)
F1, Disclosure-In-Negotiations (δ)
F21, Knew-Info-Confidential (π)
¬F4, Non-Disclosure-Agreement (δ)

Figure 1: The *Steffi v. Vincent* hypothetical problem

students typically prepare. The facts are kept simple to fit space restrictions while at the same time allowing for all techniques to be demonstrated. The description related to the Factors reflects the relevant language used in trade secret cases. This example will serve to illustrate our techniques, which we hope ultimately to apply to real case opinions.

In a written argument for the plaintiff, her lawyer would address the issues whether the information qualifies as a trade secret, whether there was a confidential relationship, and whether defendant misappropriated the information. For each issue, he would cite favorable cases that share relevant factual strengths or weaknesses with the problem.

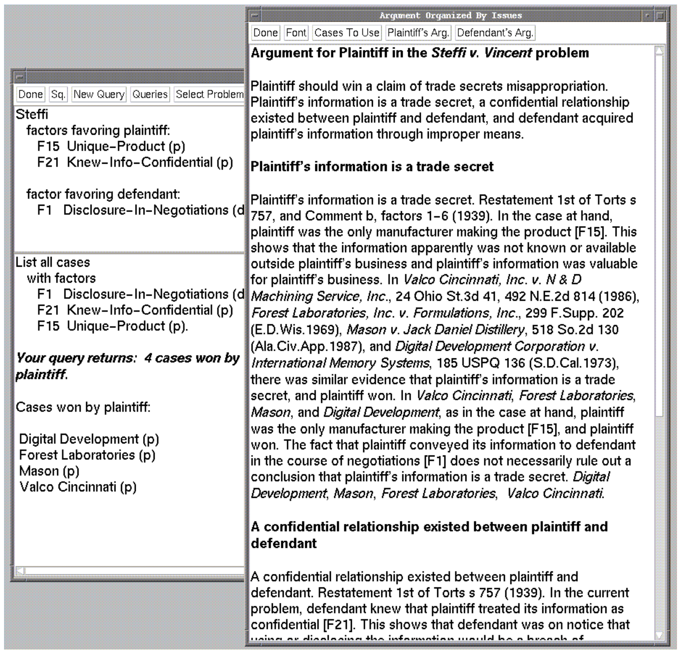
In the CATO model, such stereotypical strengths and weaknesses are represented with *Factors*. CATO is an intelligent tutoring environment for teaching law students skills of making arguments with cases (Aleven 1997). It dynamically generates arguments favoring either side's claim by analogizing or distinguishing precedents. In so doing, it compares cases in terms of its 26 Factors. Each Factor is binary (i.e., it is either present in a case or not), and favors either plaintiff or defendant. The Factor Hierarchy represents higher-level legal knowledge and relates the Factors to more abstract issues. It helps CATO organize arguments by issues and reason with partially matched cases (Ashley & Aleven 1997). The Case Database comprises 150 trade secret cases with squibs, and the full-text opinions.

In the hypothetical case, plaintiff's claim is strengthened to the degree that her method was unique, and that defendant knew the information was confidential. The corresponding sentences are italicized and marked with π and the relevant pro-plaintiff Factor from the CATO program. Defendant's position is supported by the absence of a non-disclosure agreement and by plaintiff's voluntary disclosure of information, marked with δ and the respective Factor.

Figure 2 shows how CATO could be used in the *Steffi v. Vincent* problem. In the upper left, the case is expressed in terms of CATO's Factors. The large window in the front shows CATO's argument for the plaintiff, which uses cases retrieved by a query displayed in the lower left.

3. INDEXING CASES WITH SMILE

Although empirical evidence has shown that CATO's instruction compared well with that of an accomplished legal writing instructor (Aleven 1997), CATO is not part of the law school curriculum. Similarly, despite its potential benefits for legal practice, the program has not made it to law offices yet. One of the main obstacles for a wider use is the cost associated with adding new cases to the Case Database. The law is a continuously evolving domain, where new cases are decided every day, and decisions may be overturned



Steffi
  factors favoring plaintiff:
    F15 Unique-Product (p)
    F21 Knew-Info-Confidential (p)

  factor favoring defendant:
    F1 Disclosure-In-Negotiations (d

List all cases
  with factors
    F1 Disclosure-In-Negotiations (d
    F21 Knew-Info-Confidential (p)
    F15 Unique-Product (p).

Your query returns: 4 cases won by plaintiff.

Cases won by plaintiff:

Digital Development (p)
Forest Laboratories (p)
Mason (p)
Valco Cincinnati (p)

Argument for Plaintiff in the *Steffi v. Vincent* problem

Plaintiff should win a claim of trade secrets misappropriation. Plaintiff's information is a trade secret, a confidential relationship existed between plaintiff and defendant, and defendant acquired plaintiff's information through improper means.

Plaintiff's information is a trade secret

Plaintiff's information is a trade secret. Restatement 1st of Torts s 757, and Comment b, factors 1–6 (1939). In the case at hand, plaintiff was the only manufacturer making the product [F15]. This shows that the information apparently was not known or available outside plaintiff's business and plaintiff's information was valuable for plaintiff's business. In *Valco Cincinnati, Inc. v. N & D Machining Service, Inc.*, 24 Ohio St.3d 41, 492 N.E.2d 814 (1986), *Forest Laboratories, Inc. v. Formulations, Inc.*, 299 F.Supp. 202 (E.D.Wis.1969), *Mason v. Jack Daniel Distillery*, 518 So.2d 130 (Ala.Civ.App.1987), and *Digital Development Corporation v. International Memory Systems*, 185 USPQ 136 (S.D.Cal.1973), there was similar evidence that plaintiff's information is a trade secret, and plaintiff won. In *Valco Cincinnati*, *Forest Laboratories*, *Mason*, and *Digital Development*, as in the case at hand, plaintiff was the only manufacturer making the product [F15], and plaintiff won. The fact that plaintiff conveyed its information to defendant in the course of negotiations [F1] does not necessarily rule out a conclusion that plaintiff's information is a trade secret. *Digital Development*, *Mason*, *Forest Laboratories*, *Valco Cincinnati.*

A confidential relationship existed between plaintiff and defendant

A confidential relationship existed between plaintiff and defendant. Restatement 1st of Torts s 757 (1939). In the current problem, defendant knew that plaintiff treated its information as confidential [F21]. This shows that defendant was on notice that

Figure 2: Using CATO for making arguments about the *Steffi v. Vincent* hypothetical problem

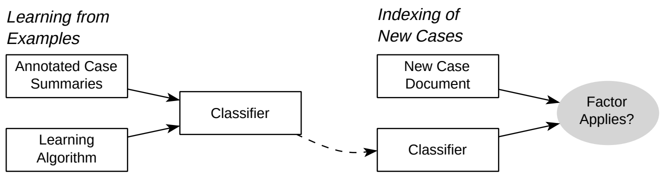


Figure 3: Overview of the learning approach and how new cases are indexed in SMILE

or questioned on appeal. Developments like these have to be incorporated into the casebase. For pedagogical reasons it is desirable to keep a tutoring system's casebase up-to-date, or to expand it to other legal domains. Our research attempts to address this problem with methods for automatically assigning Factors to new cases.

SMILE's machine learning (ML) approach was inspired by recent successful research in text classification in ML and IR (McCallum *et al.* 1998; Joachims 1998). Instead of applying advanced statistical models developed for very large sets of training examples, however, SMILE uses a symbolic ML approach; see (Thompson 2001) for a similar approach. This is more suitable for our relatively small case collection. It also facilitates the integration of background knowledge for a better representation of the training examples (Brüninghaus & Ashley 1997; 1999).

An overview of SMILE is given in Figure 3. In the *Learning from Examples* phase, SMILE takes as input a set of annotated squibs, similar to the *Steffi v. Vincent* case. The marked-up sentences are positive training instances for a Factor, and all unmarked sentences are negative instances, whether they come from a case where the Factor applies or not. The sentences are represented as a set of features. SMILE uses the symbolic learning algorithm ID3 to induce rules or a classifier from which rules can be easily extracted.

The classifier can then be used for *Indexing of New Cases*. This includes classifying sentences from squibs, like those it was trained

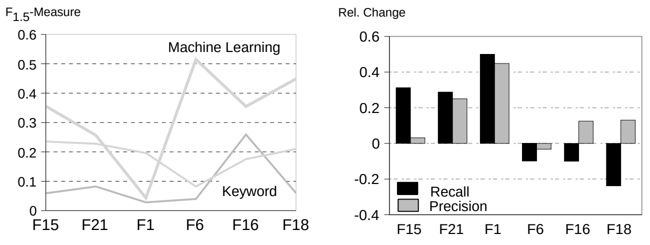


Figure 4: Results of the first evaluation of SMILE

on, and labeling sentences from the full-text opinions. The Factor is assigned to a new case text, if at least one sentence in the document was classified as a positive instance.

In initial experiments reported in (Brüninghaus & Ashley 1999), we compared classifiers learned with ID3 to two different strategies, which humans tend to use to search for Factors in case texts. Humans familiar with CATO often try to find all or at least one of the keywords from the Factors' descriptive names, see Figure 1 for some examples. Figure 4 (left) shows the results of the *Machine Learning* and *Keyword* approaches. We compared the results across Factors using the $F_{1.5}$ measure, which combines precision and recall in a single number, similar to a weighted geometric average with a slight preference for recall. Apart from the particularly difficult Factor F1, the learning algorithm scored better than the keyword strategy.

The second part of the experiment tested the integration of background knowledge in the form of an online thesaurus from WestLaw. The goal was to overcome the negative effects of overfitting. The implemented learning algorithm ID3 tends to learn overly specific trees; in particular it tries to cover all synonyms for one concept in individual branches, which can hurt performance. With the thesaurus, all variants referring to the same concept are covered by one decision tree node. Figure 4 (right) shows the relative improvements of precision and recall. We found considerable performance improvements for some of the Factors (F15, F21, and F1), effects likely to offset each other for some Factors (F16 and F18), and for one Factor (F6) a minor performance decrease.

The experiments indicate that the ML approach is promising and that adding background knowledge can improve performance. The results also show that the problem is far from being solved, and that further research has to be carried out before the classifiers learned from case squibs will measure up to the complexity of legal case opinions. A more detailed analysis of the data suggests that a more powerful representation of the examples is needed. The legal concepts underlying the Factors require a linguistic analysis of the examples and a better representation of knowledge about the language used in legal case texts.

4. TOWARD AN IMPROVED REPRESENTATION FOR LEGAL TEXT

Where large amounts of training data are available, remarkable performance can be accomplished by using advanced statistical learning algorithms. These methods typically employ a bag-of-words representation. The text is split up into single words or word pairs; word order is ignored. Recent research by West Group (Yang-Stephens *et al.* 1999) shows these statistical learning methods can be beneficial in the legal domain, for indexing case abstracts under West's Key Number classification scheme. In the West Group experiments, massive amounts of training data were available, as many as 20,000,000 already indexed abstracts.

For finding Factors, however, with much smaller numbers of training examples, the statistical learning methods did not work well (Brüninghaus & Ashley 1997). We have hundreds of examples, not millions. Moreover, assigning Factors to cases is a harder problem than finding Key Numbers. The concepts underlying the Factors are significantly more fact-specific and capture more of the contents of the individual case. The most relevant West Key Number for trade secret law, 376k10(5), is defined as "Misuse of or interference with trade secrets, inventions or patent rights", which is much more general than the Factors.

In order to classify legal texts by complex concepts using comparatively few training instances, we believe it is necessary to develop a more powerful text representation, one which allows the learning algorithm to better generalize from training examples. In the following section, we discuss how abstracting from the particular actors and events to their roles, capturing actions in more meaningful multi-word patterns and representing negation can help toward that goal.

4.1 Abstracting from Names to Roles

In legal cases, the parties, products, locations, etc., are usually referred to by their names. This is appropriate in court documents, but often prevents a classifier from generalizing from the examples, because names are specific to each case.[2] In the example sentence "Steffi tells the method to her friend Vincent.", the names of both parties, Steffi and Vincent, would at best be ignored by a learning algorithm because they do not occur in any other cases. However, if this relevant information is discarded, the sentence becomes fairly worthless as an example of Factor F1. A better solution is to preserve this information, while at the same time making sure that the unique names are replaced by more general information. Abstracting from the specific actors and events of a case to their roles in the lawsuit will alleviate this problem. CATO's squibs provide many examples of sentences where unique names obscure similarities:

- Mason disclosed part of the recipe to Randle.
- Goldberg discussed and demonstrated his screw-in lead at Medtronic's headquarters.
- Hisel sent a letter to Chrysler explaining his idea for a glass-covered holder for license plates.

With the original names, it is hard to find a general pattern in these examples. However, if we know more about the cases, we can substitute Mason, Goldberg, and Hisel with "plaintiff"; Randle, Medtronic, and Chrysler with "defendant"; and recipe, screw-in lead, and idea-for-a glass-covered-holder-for-license-plates with "information". Then, the sentences look as follows:

- Plaintiff disclosed the information to defendant.
- Plaintiff discussed and demonstrated the information at defendant's headquarters.
- Plaintiff sent a letter to defendant explaining the information.

Now, having abstracted names and substituted roles, a very important pattern becomes obvious. In each of these sentences the plaintiff discloses information to the defendant. In fact, these sentences are all examples for Factor F1 in CATO.

While trade secret opinions do not adhere to a standard format, the underlying fact situations follow a basic pattern and always contain certain elements. There are always plaintiff, defendant, and

[2]Exceptions include very common names, like Smith, or large, research-oriented companies, like DuPont, which tend to be the target of industrial espionage.

product-related information, the trade secret. This information belongs to plaintiff, and was allegedly learned in some way and used by defendant.

From an information-theoretic point of view, abstracting from names and substituting roles confers an advantage, especially where, as in SMILE, single sentences are used as examples. Substituting names by generic roles imports some of the overall case context into the examples, thereby adding information content to them.

In the first SMILE experiments, we also found empirical evidence why it is useful to substitute party names with their roles. In these experiments, overfitting was a major problem with the learning algorithm. The trees became overly sparse and unbalanced when ID3 picked (non-replaced) party or product names as attributes to distinguish positive and negative instances. This is an example of how individual names can even hurt performance.

4.2 Capturing Actions in Multi-word Features

For a human to decide whether a sentence should be a positive instance of a Factor, it is crucial to know *who* did something, or *what* was done by a particular person. Prevalent text learning methods do not support the linguistic analysis required to determine these facts and to capture them in more powerful, multi-word features. With a bag-of-words representation, there is no meaningful way to distinguish between the positive example of Factor F1, Disclosure-In-Negotiations, "Plaintiff tells her information to her friend defendant" and the negative example "Defendant told plaintiff he would treat her information as confidential." The sentences would be represented as (defendant plaintiff tell information friend) and (defendant plaintiff information tell confidential treat), respectively. The only words not shared in this representation, "friend", "confidential", and "treat", are not very decisive for the goal concept and should therefore not be considered by a classifier.

A better representation would capture the Factor's meaning: Factor F1 applies if plaintiff disclosed the information. Since one can imagine various ways of disclosing or communicating something,[3] one good feature for the positive example would therefore be (π *disclose*),[4] where the italics indicate it is present whenever a synonym of disclose is found. It captures that plaintiff was the actor, and that she communicated or disclosed something. Further good features are (*disclose* τ) and (*disclose-to* δ). The negative example would have, among others, the feature (δ *disclose*). With this more meaningful representation, it should be easier to distinguish the examples and determine whether F1 applies.

We call these features *propositional patterns* (ProPs). The ProPs only capture meaningful syntactic relationships between the words, which can be derived reliably and efficiently with a state-of-the-art NLP tool; see Section 6. We are using fairly shallow techniques and do not attempt to do natural language understanding, which involves an in-depth semantic analysis to derive a logical representation (Allen 1994). The ProPs can be generated more efficiently and presumably better support generalizing from the examples.

4.3 Dealing with Negation

Negation is another important feature for determining whether a legally relevant concept is present in a legal text. It has been shown (Riloff 1995) that small words, in particular "no" and "not", should not be removed as stopwords for text classification, because their presence can be relevant for certain concepts. These results are even more relevant for the law than for other domains, because legal discourse has an adversarial character. The court may consider evidence for and against a Factor in an opinion before presenting its conclusion. For instance, in the *National Rejectors* case, after discussing arguments presented by both parties whether the defendant knew that plaintiff's information was confidential, which is represented by Factor F21 in CATO, the court concludes "There is *no* evidence that defendant knew that plaintiff had or claimed to have any trade secrets with respect to its marketed devices." (emphasis added and names substituted)

Furthermore some Factors are defined to apply if certain facts are absent. Consider Factor, F15, Unique-Product, which applies when plaintiff's product is unique and not known in the industry. The definition of the Factor already indicates that the presence of negation can be decisive.

Negation becomes even more crucial if we also consider another Factor in CATO, F20, Info-Known-to-Competitors. This Factor represents a complementary situation to F15. When the product is not known to competitors, Factor F15 applies; when it is generally known, Factor F20 applies.

In order to correctly distinguish between examples of Factors F15 and F20, negation has to be represented in the examples. This can be illustrated by the sentences "Plaintiff's customers were well known to others in the trade.", which provides evidence for Factor F20 in *Springfield*, and "The information in the customer list was not known.", which provides evidence for F15 in *Zoecon*. The relevant difference between the sentences is the word "not".

We also found empirical evidence for the need to represent negation in our previous experiments with SMILE. For Factor F15, we analyzed for which subsets of training examples the learning algorithm could not find a good way to distinguish between positive and negative instances while it was constructing the decision tree. Quite often, the problem was that ID3 did not know about negation. We then added the words "no" and "not" to the representation and observed that the algorithm used these features in places where previously it had run out of useful features. While we did not formally evaluate these experiments, the observations suggest that adding negation is important.

4.4 Illustration: Better Classification with Improved Representation

To illustrate how a better representation could lead to better classification, consider CATO's Factor F1, Disclosure-In-Negotiations. It applies when plaintiff voluntarily disclosed the information to defendant, typically in business negotiations. In past experiments, F1 was particularly hard to find; overfitting occurred more often than for other Factors.

Figure 5 shows how an ideal (or at least close to ideal) classification tree would look. This particular tree was constructed by hand from the positive examples, including those presented in Section 4.1.

The example representation in this tree includes the improvements suggested above. First, names of and references to plaintiff, defendant and the trade-secret-related information were substituted by role identifiers π, δ, and τ, respectively. Then, the sentences were reduced to simple patterns, for instance "π disclosed τ to δ." from which the corresponding ProPs were derived. From this list, identifying four typical disclosure scenarios and constructing the classification tree in Figure 5 were straight-forward.

To show how this tree works for classifying sentences, consider a sentence from the *Steffi v. Vincent* example. "Steffi tells the method to her friend Vincent." is a positive example for the Factor F1, Disclosure-In-Negotiations. In this example, the specific names would be replaced by their roles, resulting in "π tells τ to δ." As-

[3](Rodale 1978) lists for instance reveal, tell, utter.

[4]In the remainder of this paper, we will use π, δ and τ to refer to the **p**laintiff, the **d**efendant and the **t**rade-secret-related information respectively.

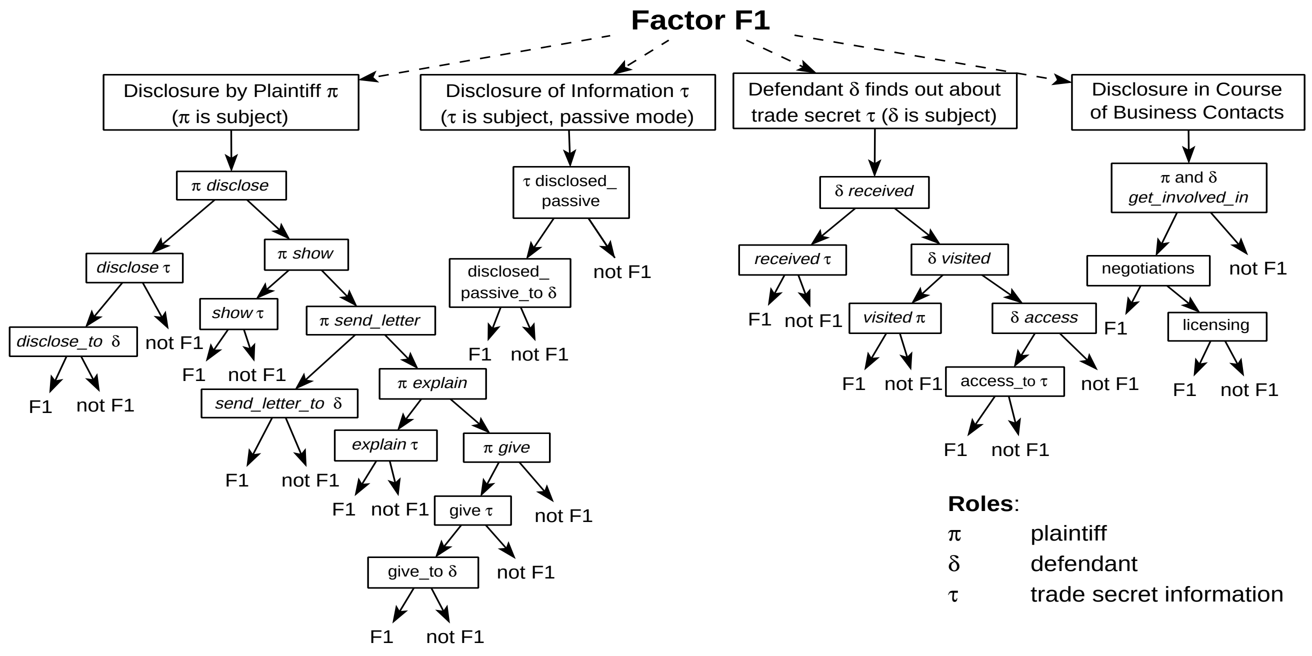


Figure 5: Manually derived classification tree for Factor F1, Disclosure-In-Negotiations

sume we have a representation of semantics which contains that the communicative act of telling something is a disclosure of what one knows.[5] Next, we can use this generalization to derive the ProPs (π *disclose*), (*disclose* τ) and (*tell-to* δ) as features for the example. Because the subject is plaintiff, we focus on the leftmost branch. The example will pass the first test, because it has the features (π *disclose*) and (*disclose* τ). Therefore, the example will be correctly classified as an instance for Factor F1. The sentence 'Vincent tells Steffi that he will treat her method as confidential." would be classified as negative. It has the subject defendant δ; we therefore focus on the third branch of Figure 5. Because this example has the feature (δ disclose), it will be handed down to the right through this branch and classified correctly as a negative instance for F1.

The resulting tree is compact and, as the example shows, fairly easy to understand. While it has not undergone a formative evaluation, this classifier can be expected to perform better than the decision trees learned in past experiments, which were much larger and harder to interpret. On unseen sentences, it should accomplish high recall. Because the features are specific for Factor F1, one can expect fewer false positives, resulting in higher precision.

5. APPLYING NLP/IE TOOLS

In order to learn the ideal classification tree discussed above, an implementation of the methods for abstracting from name to roles in the examples, for capturing multi-word features and for analyzing negation is required. While finding the parties or the product and abstracting from a sentence to word patterns is fairly easy for a human, automatically generating these features is a hard problem, because it requires background knowledge and a linguistic analysis of the documents.

Up to now, NLP techniques have been considered inappropriate for legal case texts (see (Al-Kofahi, Grom, & Jackson 1999) for a notable exception), because the language used in legal documents is too complex. Sentences in the court's opinions are exceptionally long and often have a very complex structure. Consider for instance "After a complete evidentiary hearing on plaintiff's claim for injunctive relief, the trial court concluded that Televation's schematics of its analog circuitry and the manner in which its analog circuitry interfaces with its digital circuitry are trade secrets." (from the *Televation* case). Even for an English teacher, parsing this sentence will not be easy. While many problems are still far from being solved, recent progress in NLP has yielded tools that measure up to some of the complexities of legal texts. We found that the Sundance segmenter and the AutoSlog/AutoSlog-TS IE program, developed by Ellen Riloff of the University of Utah, can handle most of the problems raised by legal cases. The following section focuses on the most important functions of AutoSlog; a more in-depth discussion can be found in (Riloff 1996).

```
CLAUSE
  NP SEGMENT (SUBJ):
    [Plaintiff (LEX)(N SINGULAR(PLAINTIFF PERSON))]

  VP SEGMENT (ACTIVE_VERB):
    [developed (root: develop) (MOR)(V PAST(MANUFACTURE))]

  NP SEGMENT (DOBJ):
    [a (LEX)(ART)]
    [method (LEX)(N SINGULAR(HOW-TO))]

  PP SEGMENT (PREP):
    [for (LEX)(PREP)]
    NP SEGMENT:
      [indexing (INF-MOR)(GER)]
      [text (LEX)(N SINGULAR)]
      [cases (root: case) (MOR)(N PLURAL)]
  [>PERIOD (LEX)(PUNC)]
```

Figure 6: Sundance's output for the sentence "Plaintiff developed a method for indexing text cases."

Given a sentence as input, the Sundance segmenter generates a (partial) parse. It has very efficient heuristics to first break up a sentence into clauses, and then find the subject, verb, object and prepositional phrases. For an example of Sundance's output, see Figure 6.

Sundance is exceptionally robust and efficient. As with all parsers,

[5]We applied this assumption in constructing the tree.

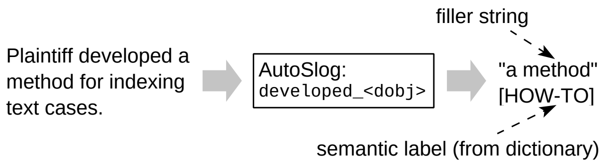

filler string
Plaintiff developed a method for indexing text cases.
AutoSlog:
developed_<dobj>
"a method"
[HOW-TO]
semantic label (from dictionary)

Figure 7: From output of AutoSlog in extraction mode

its syntactic analysis of a sentence is often not absolutely correct. However, even where the parse contains a mistake, Sundance's output is usually still useful. In particular for the long sentences in legal cases, the segmenter tends to "recover" on a later clause, which is not the case with many other parsers. A minor disadvantage of Sundance is the fact that the currently implemented heuristics were not designed for the rather long noun phrases in legal cases, which is the reason for most of the errors we encountered.

However, the most remarkable functionality of AutoSlog is its IE module. Given a target word, the program can find all linguistic contexts in which this word occurs in a document. To illustrate this, assume that the target word is "method" and that the text is "Plaintiff developed a method for indexing text cases." As we can see from the Sundance output in Figure 6, in the example sentence, "method" is the direct object. The verb of the sentence is "developed". The context for the word "method" in the example sentence is "direct-object of the verb developed". AutoSlog has heuristics to discover these linguistic contexts, called *caseframes*, from the output of Sundance. We will follow the notation in (Riloff 1996), and use `developed_⟨dobj⟩` to refer to the caseframe "direct-object of the verb developed".

AutoSlog's caseframes can be used to extract information from new texts. To illustrate how AutoSlog's extraction mode works, consider the caseframe `developed_⟨dobj⟩` ('direct-object of the word developed') and the sentence "Plaintiff developed a method for indexing text cases." As we know from Figure 6, the verb in the sentence is "developed" and the direct object is "method".

In its extraction mode, AutoSlog first checks whether the caseframe is triggered by the input sentence. That is in our example, the program tests whether the verb in the caseframe is the same as the verb of the sentence. Because both have the verb "developed", the caseframe is triggered by the sentence. In the next step, Autoslog extracts from the sentence the target indicated by ⟨⟩. In the example caseframe, the target is ⟨`dobj`⟩, the direct object. Therefore, the direct object "method" would be extracted from the input sentence. The extracted information is also called the filler.

Summarizing, given a caseframe and a sentence as input, AutoSlog extracts the target filler if the caseframe is triggered by the sentence. If the input sentence were "Plaintiff developed after determined research a computer program", "computer program" would be extracted as filler because the caseframe is triggered by the sentence.

Moreover, AutoSlog determines the semantic category of the filler. See Figure 7 for the output when the caseframe `developed_⟨dobj⟩` is triggered by the sentence "Plaintiff developed a method for indexing text cases." After the word "method" is extracted, AutoSlog infers from the semantic annotation in its dictionary that the filler is knowledge how to do something.

With this functionality, AutoSlog is a perfect tool for identifying and extracting information from CATO's case texts. The courts tend to repeat certain phrases. For instance, there are not too many ways to describe that a person, the plaintiff, developed a trade secret. Thus, from a representative training set of trade-secret-related

Training Sentence	Product	Caseframe
SDRC began developing a computer program called NIESA.	computer program	`developing ⟨dobj⟩`
	NIESA	`called:passive ⟨dobj⟩`
The plaintiff manufactures adhesive tape, including a "masking tape"	adhesive tape	`manufactures ⟨dobj⟩`
	masking tape	`including ⟨dpbj⟩`

Table 1: Examples of caseframes generated from squibs. The caseframes read: Extract ⟨syntactic-role⟩ if the sentence has the trigger `verb`.

information, like "method", together with the sentences in which the product is referred to, like "Plaintiff developed a method for indexing text cases.", one can derive a set of prototypical linguistic contexts, or caseframes, for the product, including `developed_⟨dobj⟩`. Other typical caseframes for the product are listed in Table 1. If one of those caseframes is found in a new case, it is reasonable to assume that the extracted filler information is a product.

Of course, sometimes a caseframe extracts a filler one was not looking for. Consider the sentence "Plaintiff [aka Steffi] has a friend called defendant [aka Vincent]." Table 1 contains the caseframe `called:passive_⟨dobj⟩`, which is actually very effective for extracting product names. However, this caseframe would extract "defendant", who is certainly not a product. The semantic analysis of the extracted filler from Figure 7 can help reduce these errors. It would have labeled "defendant" as `[PERSON]`. Various statistical analyses over the training set can also help detecting overly general caseframes.

6. IMPLEMENTING AND TESTING AN IMPROVED TEXT REPRESENTATION

In Section 4.1, we motivated the proposed measures for an improved text representation with three examples sentences. In the last section, we introduced the AutoSlog/Sundance NLP/IE tools. Here, we demonstrate how to construct the improved text representation by applying the tools, and report experiences and preliminary experiments with these techniques.

6.1 Finding the Parties in a Lawsuit

As mentioned above, trade secret cases always have a party who owns the trade secret, and a party who allegedly misappropriated the secret. With respect to the trade secret claim (and for CATO), they are plaintiff and defendant, respectively. If the case is an appeal, the additional roles for the appeal may appear, but they will not affect the initial trade secrets claim.[6]

To find the parties from the full-text opinions, a number of clues from the cases can be used to infer the roles of the persons. If the opinion follows a standard format, the header will read "Steffi, Plaintiff, v. Vincent, Defendant". Half of the cases in CATO's Case Database have this pattern. Here, trivial pattern matching techniques suffice. Unfortunately, the other half of the cases re-

[6]We will not consider exceptions where the parties' roles are reversed, for instance a declaratory judgement or a trade secret counter-claim within a lawsuit, which are fairly rare.

$list-of-names :	($name,)*$name(,? and $name)?
$name :	$person-name | $company-name
$person-name :	$title? $first-name? $last-name | . . .
$company-name :	$capitalized, $organization |
	$person-name & Sons | . . .
$first-name :	*read from file*
$last-name :	[A-Z][a-Z]* | [A-Z][a-Z]*-[A-Z][a-Z]*
$organization :	Inc. | Co. | Ltd. | . . .

Figure 8: Sample grammar for names. $ indicates a variable; other meta-characters have the usual meaning (Friedl 1997).

quires more work. Various courts follow different stylistic guidelines. Similarly, when the case is an appeal, the parties are often not referred to as plaintiff and defendant in the header. In the majority of these cases, the court uses appositions to ascertain the parties roles somewhere in the body of the case; for instance, "Plaintiff, Telerate Systems, Inc. ('Telerate'), ...", from the *Telerate* case. If it is possible to infer that "Plaintiff" is the same as "Telerate System, Inc." and "Telerate" in the parentheses, the names of the parties can be ascertained.

Appositions are fairly difficult to handle correctly for an NLP program, because without additional clues, their attachment and scope may be ambiguous. For our task, however, the scope of the apposition is known, because the names of the parties are given in the header of an opinion; we only need to figure out who is plaintiff and who is defendant. We also know that the head noun of the apposition is plaintiff or defendant, so we know the attachment. This makes the problem much easier to solve. We have identified and formalized three general patterns to cover appositions like in the *Telerate* example using the following sentence patterns:

- The plaintiff, `$list-of-names`, . . .
- `$list-of-names`, plaintiff, . . .
- Plaintiff `$list-of-names` . . .

These patterns are appropriate for single names and lists of names. Our implementation can also recognize potential members of a party, which were only mentioned as *et al.* in the header of the case.

We used Perl (Wall, Christiansen, & Schwartz 1996) to implement the heuristics. This language offers particularly powerful and efficient regular expressions for dealing with strings and text (Friedl 1997). In Perl, patterns for various types of names and for appositions can be expressed easily in a rule-based grammar, which also facilitates matching name variations, for instance "Mr. T. Smith" and "Tom Smith". Figure 8 is intended to give a flavor for the kind of rules and regular expressions we developed for names. These rules are not taken from the actual grammar, which is more complicated. We also use the grammar to guess whether a name refers to a person or a company.

Name recognition and matching has long been a focus in IE and IR research, and sophisticated methods have been developed, see (Thompson & Dozier 1999) for a study in the legal domain. These methods go beyond what is needed for SMILE, where the names are usually given in the opinion header. They not only have to extract personal names, but also more general information like "U.S. Fish and Wildlife Service" or "the 2000-2001 academic year" in the MUC evaluations .

In this first implementation, the heuristics work fairly well. For only a small number of cases, it is not possible to automatically determine the identity of the parties. In most of these cases, it is also quite difficult for a human to determine the roles of the parties. The only explicit evidence for identifying the parties may be

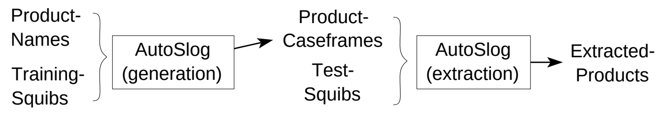


Figure 9: Experiment to extract product names and product-related information

sections where the court focuses on procedural issues. In those cases, a deeper syntactic analysis is required and AutoSlog may be used to increase coverage. In a trade secret claim, the plaintiff (by definition) sues, or brings suit against, the defendant. For instance, in the *Amoco* case, the header does not identify the plaintiff or the defendant, but we have a sentence "Amoco [plaintiff] brought suit . . . against Lindley [defendant]." Intuitively, a very indicative caseframe to determine the parties's roles is ⟨`subj`⟩`_brought-suit` (subject of a sentence that has the trigger "brought" and the direct-object "suit"), which should extract the plaintiff with very high precision. When we tested this, all fillers (apart from a few parser errors caused by punctuation) extracted with this caseframe referred to the plaintiff. While most of the fillers were names, about 40% fillers are of the form "Owner of computer software", which does not identify the name of the plaintiff. However, this filler can be used to improve upon the methods for inferring the product in this case, discussed below.

We are planning to set up a more in-depth evaluation, where we hope to achieve precision and recall comparable to the results reported in Solomon (Moens 2000) for simpler structured cases.

6.2 Finding the Product

Another technique to improve the representation is to replace the name of products and product-related information by a more general term. In this section, we discuss why finding product-related information is hard, and how we use an IE program to overcome these problems.

Finding the alleged trade secret and product-related information is very difficult. First, there may be multiple aspects to the trade secret. For instance, the product sold in the market may be a car, while the trade secret is a unique machine to produce the car, and the information stolen by the defendant were the design drawings for that machine. For the purpose of finding Factors, all these aspects of the trade secret are relevant. Second, depending on the kind of secret, the way it is referred to can vary widely. The pattern matching methods used to find the parties would not be appropriate here, a deeper syntactic analysis of the cases is necessary.

For extracting the product information, we plan to use AutoSlog. Given a set of target nouns, the program can generate the caseframes related to those nouns from input text. Examples of input sentences and product names with the corresponding caseframes are in Table 1. As mentioned above, caseframes can be interpreted as a form of extraction rule: if the trigger is present, extract the filler role. This extraction is not based on simple word matching, but on a linguistic analysis of the input case.

6.3 Evaluation and Experiments for Finding the Product

To determine whether we can use caseframes derived from a (training) set of examples to extract product information from unseen (test) cases, we conducted an experiment outlined in Figure 9. The training set comprised squibs and the product names mentioned in them. The experiment was carried out as 2-fold cross-validation (Cohen 1995).

First, we manually extracted the trade secret-related noun phrases, or Product-Names, from every squib. For each training case, we gave the respective Product-Names and Training-Squib as input to AutoSlog. The program generated the Product-Caseframes for every training case. For examples of sentences from the Squibs, the manually extracted Product-Names and the Product-Caseframes, see Table 1. Most Product-Caseframes were very useful; however, parsing errors also lead to some unwanted instances. For instance, we got a caseframe which extracts the subject triggered by the verb "signed", which will almost always return a person.

We then used the Product-Caseframes generated from the training set, as well as the Test-Squibs for the cases in the test set as input to AutoSlog in extraction mode. The outputs were the Extracted-Products. To catch some of the parser errors mentioned above, we filtered out all Extracted-Names that were automatically labeled by AutoSlog as a person, a contract or a date. To save time, we also removed extracted pronouns.

It should be pointed out that we did not delete any of the caseframes that were generated automatically from the training set. We used neither statistical thresholding nor did we manually go through the caseframes to make sure those generated by parsing errors would be removed. We only used information about the semantic category of the extracted noun phrase generated by the program where available.

From the cleaned-up list of Extracted-Products, we calculated recall as the percentage of products from a manually extracted list of trade-secret related information that were found by the program. We did not count it as a mistake when part of the noun phrase was chopped off by the segmenter. Recall was defined as the percentage of correct references to the trade secret in the set of Product-Names extracted from a squib. We scored as correct the extraction of generic terms, like machine or appliance, as well as references to defendant's products. Even though manual scoring has always a certain bias, we tried to be as objective as possible in this evaluation.

In this experiment, precision and recall, macroaveraged over the cases, reached 66.15% (σ = 0.12) and 64.82% (σ = 0.05). Given that only half of the cases were in each training set split and that we performed no manual fine-tuning or statistical analysis to detect overly general or incorrect caseframes, we feel this is a positive result. Finding product information is difficult, the language used depends on the type of trade secret, whether it is a compilation of information or a marketed product. We hope to increase precision with further measures to filter out less useful caseframes.

6.4 Generating Propositional Patterns

After the references to parties and products have been substituted with their roles in the lawsuit, propositional patterns (ProPs) can be generated. The ProPs can make the goal concepts in the textual examples more explicit. For instance, the manually constructed tree in Figure 5 has a node (π *show*), which represents sentences with "plaintiff" as subject and a synonym of "to show" as verb. This feature is powerful because it captures a relevant aspect of the target concept, Factor F1. For F1 to apply, the disclosure has to be made by plaintiff or somebody acting on his behalf.

In a first step, we used AutoSlog to generate all possible caseframes for the sentences marked-up for F1 in the squibs. Table 2 lists the most frequent patterns from the squibs related to Factor F1. It also lists the ProPs corresponding to those caseframes, which can all be found in the manually generated decision tree in Figure 5. This table provides evidence how useful the caseframes can be for deriving the targeted features.

In order to generate the ProPs from the caseframes, we will use

Caseframe	Corresponding pattern
⟨subj⟩_gain	(δ gain) [access]
⟨subj⟩_give	(π give)
⟨subj⟩_show	(π show)
letter_to_⟨pp⟩	(letter_to δ)

Table 2: Most frequent caseframes from examples for F1 with the corresponding ProPs

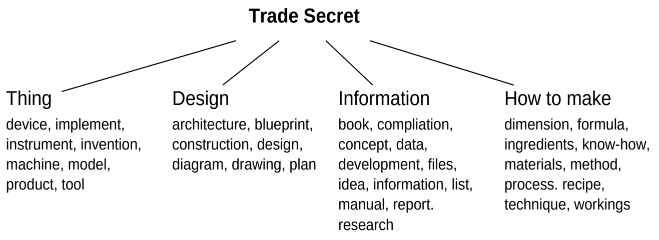

Trade Secret
Thing
device, implement, instrument, invention, machine, model, product, tool
Design
architecture, blueprint, construction, design, diagram, drawing, plan
Information
book, compliation, concept, data, development, files, idea, information, list, manual, report. research
How to make
dimension, formula, ingredients, know-how, materials, method, process. recipe, technique, workings

Figure 10: Semantic hierarchy for trade-secret-related information

those caseframes and extract the filler noun phrases from the example sentences. Then, we will generalize from the fillers to more general concepts, using the semantic analysis performed by AutoSlog. In Figure 7 for instance, the semantic label [HOW-TO] for the filler "method" was inferred automatically by the program. We are in the process of representing the relevant language in a semantic hierarchy, so that it can be used to determine the best semantic label for the filler by AutoSlog. Figure 10 shows part of the semantic hierarchy for trade-secret-related information, which we derived combining our analysis the cases and from a legal thesaurus (Burton 1992; Statski 1985).

6.5 Determining the Scope of Negation

The last technique to improve the representation of examples for finding Factors is the detection and representation of negation.

Negation is one of the harder problems in NLP. The presence or absence of the words "no" or "not" in a sentence is not equivalent with the presence or absence of negation. Furthermore, even if "no" or "not" are spotted, this does not indicate what was negated. To illustrate why the scope of negation is important, consider a sentence from the *Steffi v. Vincent* example, "Because they were friends, the defendant did not sign a non-disclosure agreement." For deciding that this sentence should not be classified under Factor F4, Non-Disclosure-Agreement, it is important to know what is negated. Only a linguistic analysis of the sentence can determine the scope of the negation. Consider a variation of the example, where we move the "not" five words to the left: "Because they were not friends, the defendant did sign a non-disclosure agreement." This sentence has to be classified under F4. Unless we can ascertain not only that there is negation, but what is negated, it is impossible to correctly classify the two examples.

Our proposed solution is to use Sundance to determine the likely scope of the negation in the examples. The Sundance segmenter reliably splits a sentence into clauses before analyzing it syntactically. The example would be split into the clauses "they were friends" and "the defendant did not sign a non-disclosure agreement". These clauses are the typical scope of negation in the squibs. Therefore, we can assume that all features, in particular the ProPs, derived from a clause which contains a negation are negated.

Currently, we are only focussing on simple instances of negation. Depending the results of the planned experiments, we will continue to explore indirect speech and expressions of probability and doubt.

7. SUMMARY

In this paper, we have reported on our research toward automatically indexing legal case texts using ML methods. We have argued that existing methods, both for extracting information from legal texts and for document classification are not powerful enough for finding indexing concepts in cases. From the analysis of our past experiments, we identified three hypotheses how a more powerful representation of the text cases will lead to improved indexing.

We predict that a learning algorithm will better generalize from examples and classify new cases if the representation (1) abstracts from the specific names of the parties and products to their roles in the lawsuit, (2) captures events and actions in multi-word features, and (3) recognizes negation. In a hand simulation, we showed how these methods can lead to a more powerful classifier. We have presented the design for an implementation, and evaluated the first modules. The experiments suggest that domain-specific heuristics are appropriate for finding the parties and that a state-of-the-art IE tool can be applied to detect the product. We are continuing to implement the outlined methods for deriving multi-word features and representing negation.

If successful, the methods presented here could be beneficial for other applications, beyond case indexing. For instance, if the cases in a full-text retrieval system were represented as suggested here, more abstract and powerful queries and quasi-conceptual retrieval would be possible. For instance, instead of submitting a query for "disclos*" (all words beginning with disclos) and "plaintiff", which would return almost every trade secret case, one could ask for cases where plaintiff did disclose something, that is, for cases with the ProP (π disclose).

It has long been recognized that a better text representation could lead to significantly improved information retrieval systems (Turtle 1995), yet all approaches that integrated NLP tools to derive phrases or used thesauri to find synonyms did not lead to the expected, consistent performance improvements; see for instance (Voorhees 1998). We believe that the improved text representation outlined above is more likely to lead to improvements than the domain-independent methods tried in the past, which were often too general. The methods discussed here were developed specifically for legal cases and for the kind of concepts relevant in the legal domain.

ACKNOWLEDGEMENTS

We are very grateful to Ellen Riloff from the University of Utah for letting us use her AutoSlog program. We also thank Vincent Aleven from Carnegie Mellon University for his comments on this paper and for being a friend who would never missapropriate a trade secret. This research has been supported by the National Science Foundation under Grants IRI96-19713 and IRI99-87869.

REFERENCES

Al-Kofahi, K.; Grom, B.; and Jackson, P. 1999. Anaphora resolution in the extraction of treatment history language from court opinions by partial parsing. In *Proceedings of the Seventh International Conference on Artificial Intelligence and Law*, 138–146.

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

Allen, J. 1994. *Natural Language Understanding*. Benjamin/Cummings Publishing.

Ashley, K., and Aleven, V. 1997. Reasoning Symbolically about Partially Matched Cases. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, 335–341.

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

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*, 303–314.

Brüninghaus, S., and Ashley, K. 1999. Toward Adding Knowledge to Learning Algorithms for Indexing Legal Cases. In *Proceedings of the Seventh International Conference on Artificial Intelligence and Law*, 9–17.

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

Cohen, P. 1995. *Empirical Methods for Artificial Intelligence*. MIT Press.

Daniels, J., and Rissland, E. 1997. Finding Legally Relevant Passages in Case Opinions. In *Proceedings of the Sixth International Conference on AI and Law*, 39–46.

Friedl, J. 1997. *Mastering Regular Expressions*. O'Reilly & Associates, Inc.

Joachims, T. 1998. Text Categorization with Support Vector Machines: Learning with many relevant Features. In *Proceedings of the European Conference on Machine Learning (ECML-98)*.

McCallum, A.; Rosenfeld, R.; Mitchell, T.; and Ng, A. 1998. Improving Text Classification by Shrinkage in a Hierarchy of Classes. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 359–367.

Moens, M.-F. 2000. *Automatic Indexing and Abstracting of Document Texts*. Kluwer Academic Publishers.

Prakken, H., and Sartor, G. 1998. Modelling Reasoning with Precedents in a Formal Dialogue Game. *Artificial Intelligence and Law* 6:231–287.

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*.

Riloff, E. 1996. Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1044–1049.

Rissland, E.; Skalak, D.; and Friedman, T. 1996. BankXX: Supporting Legal Arguments through Heuristic Retrieval. *Artificial Intelligence Review* 10:1–71.

Rodale, J. 1978. *The Synonym Finder*. Warner Books.

Statski, W. 1985. *West's Legal Thesaurus and Dictionary*. St. Paul, MN: West Publishing.

Thompson, P., and Dozier, C. 1999. Name Recognition and Retrieval Performance. In Strzalkowski, T., ed., *Natural Language Information Retrieval*. Kluwer Academic Publishers. 261–272.

Thompson, P. 2001. Automatic Categorization of Case Law. In *Proceedings of the Seventh International Conference on AI and Law*.

Turtle, H. 1995. Text Retrieval in the Legal Word. *Artificial Intelligence and Law* 2(1):5–54.

Voorhees, E. 1998. Using WordNet for Text Retrieval. In *WordNet: An Electronic Lexical Database*. MIT Press. 285–303.

Wall, L.; Christiansen, T.; and Schwartz, R. 1996. *Programming Perl, 2nd Ed.* O'Reilly & Associates.

Weber, R. 1999. Intelligent Jurisprudence Research: a new concept. In *Proceedings of the Seventh International Conference on AI and Law*, 164–172.

Yang-Stephens, B.; Swope, M.; Locke, J.; and Moulinier, I. 1999. Computer-Assisted Classification of Legal Abstracts. In *Proceedings of the Third International Symposium on Advances in Intelligent Data Analysis*, 437–448.