Named Entity Recognition and Resolution in Legal Text

Conference Paper · January 2010

DOI: 10.1007/978-3-642-12837-0_2 · Source: DBLP

6 authors, including:

Christopher Dozier
Thomson Reuters
14 PUBLICATIONS **173** CITATIONS
SEE PROFILE

Ramdev. Wudali
Thomson Reuters
3 PUBLICATIONS **89** CITATIONS
SEE PROFILE

Named Entity Recognition and Resolution in Legal Text

Christopher Dozier, Ravikumar Kondadadi, Marc Light,
Arun Vachher, Sriharsha Veeramachaneni, Ramdev Wudali

Thomson Reuters Research and Development
Eagan, MN 55123, USA
{chris.dozier,ravikumar.kondadadi,marc.light,arun.vachher,harsha.veeramachaneni,ramdev.wudali}@thomsonreuters.com
http://www.thomsonreuters.com

Abstract. Named entities in text are persons, places, companies, etc. that are explicitly mentioned in text using proper nouns. The process of finding named entities in a text and classifying them to a semantic type, is called named entity recognition. Resolution of named entities is the process of linking a mention of a name in text to a pre-existing database entry. This grounds the mention in something analogous to a real world entity. For example, a mention of a judge named *Mary Smith* might be resolved to a database entry for a specific judge of a specific district of a specific state. This recognition and resolution of named entities can be leveraged in a number of ways including providing hypertext links to information stored about a particular judge: their education, who appointed them, their other case opinions, etc.
This paper discusses named entity recognition and resolution in legal documents such as US case law, depositions, and pleadings and other trial documents. The types of entities include judges, attorneys, companies, jurisdictions, and courts.
We outline three methods for named entity recognition, lookup, context rules, and statistical models. We then describe an actual system for finding named entities in legal text and evaluate its accuracy. Similarly, for resolution, we discuss our blocking techniques, our resolution features, and the supervised and semi-supervised machine learning techniques we employ for the final matching.

Key words: named entity recognition, named entity resolution, natural language processing

1 Introduction

Names are proper nouns and, in English, are usually capitalized and have other syntactic characteristics that differentiate them from other types of nouns. For example, names are often used without a determiner: *Thomas wrote in his dissent that the majority argument ...* Semantically, names can be thought to refer to a single entity in the world. Names play a central role in the information content of

many types of human language, including legal texts: names are used to identify parties, attorneys, courts, jurisdictions, statues, judges, etc. involved in a legal proceedings. Thus, being able to recognize which word sequences are names and resolve these names to what they refer is useful for many legal text processing tasks. In this paper we describe methods for named entity (NE) recognition and resolution in legal texts.

To make this discussion more concrete, consider the text in Figure 1.

SUPREME COURT OF THE UNITED STATES
Syllabus
MICROSOFT CORP. v. **AT&T CORP.**
CERTIORARI TO THE UNITED STATES COURT OF APPEALS FOR THE FEDERAL CIRCUIT
No. 051056. Argued February 21, 2007—Decided April 30, 2007
It is the general rule under **United States** patent law that no infringement occurs when a patented product is made and sold in another country. There is an exception. Section 271(f) of the **Patent Act**, adopted in 1984, provides that infringement does occur when one "suppl[ies] . . . from the United States," for "combination" abroad, a patented invention's "components." 35 U. S. C. §271(f)(1). This case concerns the applicability of §271(f) to computer software first sent from the **United States** to a foreign manufacturer on a master disk, or by electronic transmission, then copied by the foreign recipient for installation on computers made and sold abroad. **AT&T** holds a patent on a computer used to digitally encode and compress recorded speech. **Microsoft**'s **Windows** operating system has the potential to infringe that patent because **Windows** incorporates software code that, when installed, enables a computer to process speech in the manner claimed by the patent. **Microsoft** sells **Windows** to foreign manufacturers who install the software onto the computers they sell. **Microsoft** sends each manufacturer a master version of **Windows**, either on a disk or via encrypted electronic transmission, which the manufacturer uses to generate copies. Those copies, not the master version sent by **Microsoft**, are installed on the foreign manufacturer's computers. The foreign-made computers are then sold to users abroad.

Fig. 1. Example legal text

Names are used to refer to the companies Microsoft and AT&T, the product Windows, the location United States, the court Supreme Court of the United States, and the section of U.S. Code called The Patent Act. Resolving such entities to particular entries in lists may be straightforward as in the case of the *Supreme Court of the United States* or might be more involved as in the *AT&T* because of the numerous companies related to AT&T (e.g., AT&T Wireless). *The Patent Act* is also ambiguous: there are many different versions of the Patent Act.

Humans are able to spot such names and disambiguate their references; this ability is part of understanding a legal text. For machines, recognizing and resolving names is an initial step towards making the semantics of the legal text

explicit and available for further processing. Further processing might include placing ID numbers resulting from resolution in search engine indexes. Users can then search for specific entities instead of words. The user interface would need to provide a way to specify an ID. One easy way to enable this search is to allow a user to initiate a search on an entity by clicking on it in a document, e.g., clicking on *AT&T* in Figure 1 would start a search for other documents that mention this specific company. Alternatively, such mentions of names in text could be hyperlinked to renderings of the corresponding database information. This landing page might have links out to other documents mentioning the entity. Another use would be to gather entity-based summary information such as how many cases has Microsoft been a defendant in this past year or what the law firms were, defending AT&T in the U.S. first circuit federal court system.

In the remainder of this article we present a number of methods for recognizing named entities and describe a system based on these methods. We then discuss the resolution task and again describe an actual system.

2 Named Entity Recognition

We use three methods for NE recognition: lookup, pattern rules, and statistical models. These methods can also be combined in hybrid systems.

The lookup method consists of creating a list of names of the entities of interest, such as drug names, and then simply tagging all mentions of elements in the list as entities of the given type. For example, if Vioxx is in the list of names and it appears in a document, then mark it as a drug name. Drug names are often unusual and thus often unambiguous: if they appear in text, then, with a high degree of certainty, the words refer to the drug. In contrast, a list of judge names would contain many names that are common in the greater population and thus NOT unambiguously judge names. Often such common names can be weeded out to ensure that the list reaches an acceptable level of unambiguousness. For example, using U.S. government census data, the commonness of names can be estimated. The advantages of the lookup approach are that it is simple to implement and maintain, it can make use of preexisting lists, it will find names regardless of context, and it does not require any training data. The disadvantages are that it may generate many false positives,[1] if the list contains

[1] A false positive is a situation where the NE recognizer believes that a name exists but it does not. For example, a lookup tagger might propose that *Bush* refers to George W. Bush in *My Life in the Bush of Ghosts was recorded in 1981 with Brian Eno*. A false negative is a name that is passed over by the tagger, e.g., perhaps *Brian Eno* in the sentence above. True positives are those names that the tagger finds and true negatives are those non-names that are correctly ignored by the tagger. Two measures of accuracy can be defined using counts of these classifications. Precision is the number of true positives divided by the sum of the counts of true and false positives. Recall is true positives divided by the sum of true positives and false negatives. Precision measures how often the guessed entities are correct and recall measures how often the system finds the entities that are actually there.

many ambiguous words. It may also make a number of false negatives if the list is not comprehensive enough. The basic problem is that lookup taggers ignore contextual cues for names and their types. An example cue would be that person names usually follow *Mr.*

Contextual rules encode such cues in deductive rules, e.g., if a capitalized word sequence follows *Mr.*, tag it as a person name. By looking at development data, a knowledge engineer can develop a set of such rules that recognizes the majority of the instances in the data and does not produce many false positives. The advantages of this approach is that often quite high accuracy can be obtained. The disadvantages are that the development of such rules requires manually annotated development data and often a large amount of effort from experienced rule writers. The development data is required so that the rule writers have examples to build rules from, examples to test new rules, and examples to do regression testing against. In addition to having enough manually annotated development data, it is also important for this data to be representative: ideally it should be a random sample of the data that the system will see at runtime. Maintenance of such rule sets can be a challenge, since often the rules have intricate inter-dependencies that are easy to forget and make modification risky. Finally, many descriptions of cues involve words such as usually or sometimes. Providing some sort of weighting on the rules can encode such partial aspects. However, how to set these weights optimally is often a difficult and time consuming task and further complicates maintenance.

Statistical models offer an alternative to contextual rules for encoding contextual cues. One way of thinking about such statistical models is as a set cues that receive weights and whose weights are combined based on probability and statistical concepts. A knowledge engineer must develop features that correspond to cues, pick the appropriate statistical model, and train the model using training data. As with contextual rules, statistical models can achieve high accuracy. The disadvantages are that the development of such models requires manually annotated training data and often a large amount of effort from an experienced machine learning expert. The training data is the same sort of data as the development data mentioned above: it needs to be representative of the data that the model will process at runtime. In our experience, the amount of manually annotated data required for writing good contextual rules is very similar to the amount of data needed to train a good statistical model. However, maintenance can be easier than with contextual rules in that only more hand annotated data is required to change a model's behavior.

Each of these three approaches has its place in the toolkit of an engineer trying to build named entity recognizers for legal text and we will see examples of each in the system we describe below. In addition, the methods can be combined in a number of ways. See [6] for a discussion of how best to combine such systems. We need to point out that we did not invent the three methods above; in fact, each has a long history in the named entity recognition literature. There has been a wealth of work done on named entity recognition in the natural language processing community but we will not try to review it here nor provide an

exhaustive bibliography. Instead we will direct the reader to the following review articles: general review [**?**], review of work on newswire text [8], review of work in bio-health domain [7]. We are aware of no work outside our own on legal texts: see [1, 3].

2.1 A named entity recognition system for legal text

Legal documents can be classified as captioned/non-captioned. Captioned documents have extensive header material preceding the body of the document, which usually includes attorney information, court & jurisdiction, parties (plaintiffs and defendants), and document title. Captions may also list the primary judge of the case and an identification number for the case. There are also litigation documents without the caption information but we will focus on captioned documents here. Figure 2 shows a sample captioned document. All named entity recognition systems described in this section except the judge tagger work with captioned documents.

Preprocessing: Before a captioned document is processed by the named entity tagger, we apply the following preprocessing : tokenization, zoning, and line-blocking.

Tokenization is the process of breaking up text into its constituent tokens. Our tokenization also involves identifying the line breaks in the text. Court, judge, and jurisdiction taggers work at the token level and the others use lines as input.

The zoning task involves identifying the caption portion of the legal document. Our zoner uses a statistical model (a conditional random field [9] which was trained on a manually-tagged training set of around 400 documents. The features used can be classified into the following categories:

- N-gram features: frequent n-grams occurred around the zone in the training data,
- Positional features: position of the line in the document like the first quarter, last quarter, first-tenth etc.,
- Punctuation features: punctuation like dashed lines are good indicators of the beginning of a new zone.

Line blocking is the process of identifying structural blocks of text in a document. It groups contiguous lines of the document into meaningful blocks. We developed a rule-based system based on formatting and case information to identify the blocks.

Jurisdiction Tagger: Jurisdiction refers to a particular geographic area containing a defined legal authority. Jurisdiction can be Federal, State or even smaller geographical areas like cities and counties. Jurisdiction is usually tied to the court. In order to find the jurisdiction, the jurisdiction tagger performs a longest substring match on the court context. Court context is the 5 line window surrounding the first line in the document containing the word *court*.

Adam B. Eve (2312)
Max M. Payne (2721)
Swinburne Wilde & Larkin
201 Main Street, Suite 23
Salt Lake City, Utah 84321-1219
Telephone: (801) 555-5555
Facsimile: (801) 555-5556
Ming Chang*
Joseph Stalin*
Lincoln Jefferson & Roosevelt, PC
One Cross Street
Oconomowoc, Wisconsin 34233-3423
Telephone: (334) 555-2009
Facsimile: (334) 555-2002
Attorneys for Defendants Clark Kent,
Jackie Chan and CrouchingTiger Solutions, Inc.
**Admitted Pro Hac Vice*

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF WYOMING
CENTRAL DIVISION

JUSTICELEAGUE COMMUNICATIONS, INC., a Utah corporation, Plaintiffs, vs. CLARK KENT, an individual, JACKIE CHAN, an individual, CROUCHINGTIGER SOLUTIONS, INC., a Rhode Island corporation, and STINGRAY SYSTEMS, CORPORATION, INC., an Mississippi corporation, Defendants.	**Memorandum In Support of partial Motion for Summary Judgment** Filed Under Seal Civil No. 2:09xx00065TC Honorable James Dredd Magistrate Ray Robinson

Introduction

It is important in any trade secrets case to determine whether the Plaintiff owns any trade secrets and, if so, to define the scope of such secrets. Defendants Jackie Chan, Clark Kent and CrouchingTiger Solutions, Inc. file this Partial Motion for Summary Judgment because Plaintiff JusticeLeague Communications, Inc. has admitted on Rule 32(b)(7) deposition that is does not have any admissible evidence that the GBear Code that it alleges was copied

Fig. 2. A captioned legal document.

Court Tagger: Court tagger is a lookup tagger. It extracts different components of the court like the jurisdiction, division and circuit from the court context and looks for a court in the authority database with those constituents.

Title tagger: We used a statistical approach to extract titles from the document. A title can span more than one line in the document. So as a predecessor to the title tagger, we apply line blocking to group adjacent lines into blocks. The title classifier classifies each block as title or not. Features for the title classifier include case information, formatting, position, and length. We used a manually annotated data set of 400 documents to train the classifier.

Doctype tagger: A legal document can be assigned to a category based on its contents (its doctype). Some example doctypes are Brief, and Memorandum. The doctype is usually a part of the title of the document. Once the title of the document is found, the doctype tagger performs a longest substring match on the title to find the doctype from the doctype list.

Judge Tagger: The judge tagger looks for honorifics such as *Honorable, Hon., Judge* in the document and extracts names that follow them as judges. We also developed a similar attorney tagger. These taggers are prototypical context rule taggers.

Table 1 lists the precision and recall for all the taggers. The test set includes 600 documents randomly selected from a large collection of legal documents.

Table 1. Precision and Recall of Taggers

Tagger	Precision	Recall
Jurisdiction	97	87
Court	90	80
Title	84	80
Doctype	85	80
Judge	98	72

2.2 Summary of named entity recognition

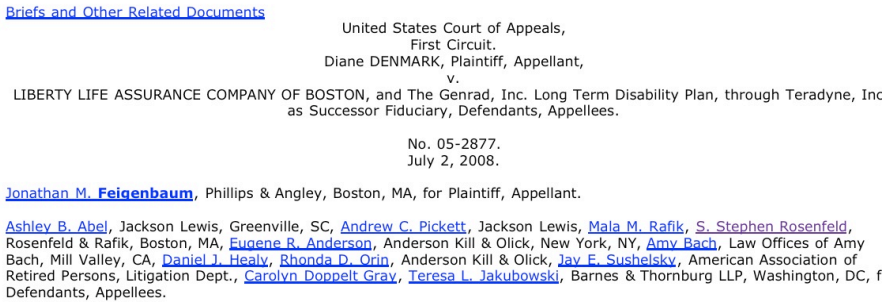
In this section, we have introduced the named entity recognition task and described three approaches to performing it: lookup, context rules, and statistical models. We then presented a system that recognizes a number of named entities types in captioned legal documents. In the next section, we move on to named entity resolution.

3 Named Entity Resolution

The job of entity resolution is to map the names with class labels to particular entities in the real world by assigning them to entity records within a class authority file.

A class authority file is a file in which each record identifies a particular person or entity in the real world belonging to the class. Examples of authority files pertinent to the legal domain include files for attorneys, judges, expert witnesses, companies, law firms, and courts.

As mentioned above resolution enables a number of applications. Thomson West's Profiler is an example of an application that allows hyperlinking from names in text to text pages describing the named entity [1]. In this system, attorneys, judges, and expert witnesses in caselaw documents, dockets, pleadings, and briefs are tagged and linked to the curriculum vitae of particular legal professionals. A screen shot showing a caselaw document with attorneys and judges highlighted is shown in figure 2.


Briefs and Other Related Documents

United States Court of Appeals,
First Circuit.
Diane DENMARK, Plaintiff, Appellant,
v.
LIBERTY LIFE ASSURANCE COMPANY OF BOSTON, and The Genrad, Inc. Long Term Disability Plan, through Teradyne, Inc as Successor Fiduciary, Defendants, Appellees.

No. 05-2877.
July 2, 2008.

Jonathan M. **Feigenbaum**, Phillips & Angley, Boston, MA, for Plaintiff, Appellant.

Ashley B. Abel, Jackson Lewis, Greenville, SC, Andrew C. Pickett, Jackson Lewis, Mala M. Rafik, S. Stephen Rosenfeld, Rosenfeld & Rafik, Boston, MA, Eugene R. Anderson, Anderson Kill & Olick, New York, NY, Amy Bach, Law Offices of Amy Bach, Mill Valley, CA, Daniel J. Healy, Rhonda D. Orin, Anderson Kill & Olick, Jay E. Sushelsky, American Association of Retired Persons, Litigation Dept., Carolyn Doppelt Gray, Teresa L. Jakubowski, Barnes & Thornburg LLP, Washington, DC, f Defendants, Appellees.

Fig. 3. Caselaw document displayed in Westlaw with attorney names highlighted. Users may jump directly to attorney's curriculum vitae by clicking on name.

The Profiler application also shows how NE resolution enables indexing of all references to particular entities within text collections [2]. In Profiler, a user can find all the cases, briefs, and settlements in which a particular attorney has been mentioned. An example of the index display for an attorney is shown in figure 3.

An example of a system that is built upon extracted relationships between resolved entities is Thomson West's Litigation Monitor [3]. For the Litigation Monitor, we extracted and resolved judges, attorneys, and companies in caselaw documents and dockets throughout the federal and state legal systems. We took the additional step of linking the company references to their representing attorneys. By combining these resolutions, Litigation Monitor can show all the companies a particular attorney has represented in federal and state court as well as all the attorneys a particular company has used for litigation. By combining this information with firm, court, judge, legal topic, time information, and

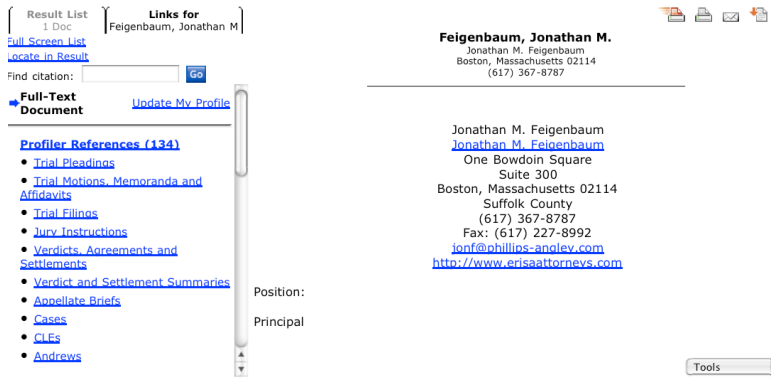


Fig. 4. Profiler page for attorney Jonathan Feigenbaum. Page shows curriculum vitae of Mr. Feigenbaum and also lists all cases, briefs, motions, pleadings, and other documents mentioning Mr. Feigenbaum.

role information (i.e., is a company the defendant or the plaintiff?), Litigation Monitor serves as a robust utility that can display litigation trends across multiple informational axes. Figure 4 shows all the law firms that have represented the Microsoft Corporation in federal and state court.

3.1 Record Linkage Approach to Named Entity Resolution

The mapping to an authority file approach to NE resolution involves a two step process: information extraction and record linkage. The extraction part consists of moving a tagged name from a given semantic class and information co-occurring with the name into a structured record. The record linkage part involves matching the structured record to a record within an authority file.

The subtasks we need to undertake to build a NE resolution system include the following.

The first task is to analyze the authority files inherent ambiguity. The purpose of this task is to determine which sets of record attributes uniquely identify a record in the authority file.

The second task consists of designing a text structured record that includes enough fields that an unambiguous match to the authority can be made in most cases. A text structured record is a structured record that contains information about a named entity tagged in text. For example, a text structured record for an attorney in a caselaw record would include the first, middle and last name of the attorney taken from the tagged attorney name in text and it might also include names of cities, states, and law firms that co-occur in the same paragraph with the attorney name.

The third task involves manually matching a relatively small set of text structured records to authority file records. This set of record pairs is called the test match set and is used as a gold standard against which to measure precision

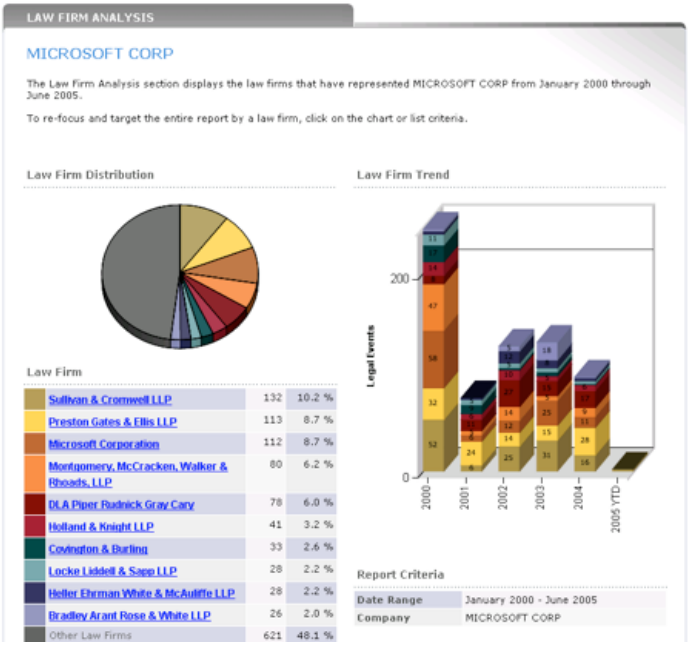


Fig. 5. This Litigation Monitor displays the law firms that have represented the Microsoft Corporation in federal and state courts between the year 2000 and the beginning of 2005.

and recall of the NE resolution system. We also use the test set to evaluate record blocking strategies.

The fourth task is to design a blocking strategy for retrieval of candidate authority file records to match against a text structured record. We use the test match set to guide the blocking strategy selection.

The fifth task consists of developing a set of match feature functions to compare candidate authority file record fields to text structured record fields. The output of these feature functions will be used to create a match feature vector. The matching classifier accepts as input the match feature vector associated with an authority-text structured record pair and returns a match belief score. The match belief score indicates how likely it is that a text structured record and authority file record pair refer to the same real world entity.

The sixth task involves creating a set of training data by matching a set of text structured records to authority file records. This step is similar to step 2 except that we need a larger number of training pairs than test pairs and in some cases we can create training data automatically using surrogate training features.

The seventh task is to train a match belief scorer using the training data. The scorer can be based on a variety of machine learning techniques. In this paper, we will discuss using a support vector machine to train a scorer for matching attorney names in caselaw to an attorney authority file.

Ant the final task is to deploy structured record creation, blocking, feature function processing, and match belief scoring in a pipe line to resolve text entities tagged in text to authority file records. We assign entity ids from authority record to text names to create text hyperlinks and indexes.

In the following sections, we discuss each of these steps and illustrate them with the example of matching attorney names in caselaw to an authority file of attorney records. In this application, we tagged the attorney names, cities, state, and law firm names in 43,936 U.S. federal cases. From the representation paragraphs in these cases, we extracted text structured records for each attorney name and matched the attorney names to an authority file of U.S. attorneys. A representation paragraph is a paragraph in which the attorneys representing the parties to the case are listed. A representation paragraph is shown in figure 5. The text structured record for *Mark D. Stubbs* is shown in table 1.

Mark D. Stubbs, Barnard N. Madsen, Matthew R. Howell, Fillmore Spencer LLC, Provo, UT, Ashby D. Boyle, II, Fillmore Spencer, Sandy, UT, for Plaintiffs.

Fig. 6. Example of Legal Representation Paragraph

Table 2. Example Text Structured Record.

Record Field Label	Field Value
first name	Mark
middle name	D
last name	Stubbs
name suffix	null
city-state name	Provo,UT:Sandy,UT
firm name	Fillmore Spencer LLC

3.2 Analyzing Ambiguity of Authority File

The ambiguity of an authority file means the degree to which the entities in the authority file have overlapping identifying information. For example, if a large percentage of the attorneys in the attorney authority file have the same first and last name, then the ambiguity of this file would be high with respect to these two attributes.

A useful way to assess the ambiguity of a file is to measure the mean number of records that share the same set of attribute settings for any given record in the file. A large mean number of records for a particular set of attribute values means that this set of attributes on its own is not likely to produce good resolution accuracy. A mean cluster size very close to 1.0 for a set of attributes, on the other hand, is likely to produce good resolution accuracy (provided the authority file is fairly well populated).

One can discover viable attribute sets to use for name resolution matching by measuring the ambiguity of different combinations of attributes.

Table 2 shows an analysis of the ambiguity of different attribute combinations for attorneys in our attorney file. The attorney authority file lists over one million U.S. attorneys. The identifying attributes for an attorney include first, middle, and last name as well as city, state, and firm name.

We can see from table 2 that on average a given first and last name combination is associated with 4.02 authority file records. So a record linkage system that relied only on these fields to make a match would not be very accurate. On the other hand, combining first and last name with firm, location, or middle name information would yield accurate results. Note that we would expect record linkage accuracy to text structured records to be lower than accuracy predicted by our ambiguity analysis because information in the text structured record may be incorrect due to extraction errors or may be out of sync with information in the authority, as when an attorney changes firms or even last names due to marriage. However, our ambiguity analysis does show what kind of accuracy we could expect with perfection extraction and synchronization of data between text and authority file.

Table 3. Mean Number of Records Sharing Field Set Attributes in Attorney Authority File.

field sets	mean number records
first+last	4.0193
first+last+state	1.2133
first+last+city+state	1.0718
first+last+firm	1.0454
first+last+firm+state	1.0175
first+last+middle	1.000

3.3 Designing Text Structured Record

The text structured record is a record whose fields are populated with information extracted from text that pertains to a particular tagged named entity. The fields of the text structured record should align semantically with the fields in the authority file. The set of fields in the text structured record should encompass enough information that an unambiguous match can be made to an authority file record when the text fields are populated. The text structure record typically will include the name of the tagged entity as well as other co-occurring information that corresponds to the attributes in the authority file record. Our analysis of the ambiguity of the authority file can indicate to us which sets of attributes we should incorporate into our text structure record.

3.4 Developing Test Data

To test an NE resolution system, we need a set of pairs of authority-text structured records that refer to the same individual (positive instances) and a set of pairs of authority-text structure records that do not refer to the same individual (negative instances). We measure the precision of our NE resolution system by dividing the number of correct positive matches made by the total number of matches made. We measure recall by dividing the number of correct positive matches made by the number of positive instances manually identified.

Since the authority file should contain at most one record that truly matches a named reference in text, we can automatically create many negative test examples from each positive authority-structured record pair by assuming that all authority-structured record pairs derived from a retrieved block of candidate records are negative (i.e. do not co-refer) except the positive (i.e. co-referring) pair which has been identified.

One method of creating test data is to randomly select structured records extracted from the text and manually match them to records in the authority file. Typically we want to have at least 300 positive test record pairs. For our example attorney NE resolution system, we created 500 manual matches between attorney names mentioned in our database of U.S. federal cases and attorneys listed in our attorney authority file.

Table 4. Blocking Efficiency for Three Different Block Keys.

block key	recall	mean block size
last name	1.0	403.2
last name+first init	0.97	7.6
last name+first name	0.95	4.4

3.5 Selecting Blocking Functions

The purpose of blocking is to increase the efficiency of an NE resolution system by limiting the number of authority records to which we must compare the text structured record in order to find a match. A good blocking function will return a small mean number of authority records for text structured records under consideration and will also return with a high probability the authority record that best matches each text structured record.

We can use our test data to assess different blocking functions. In table 3, we show the blocking efficiency of three different blocking keys. To compute these numbers, we constructed a text structured record from text associated with each of our 500 test instances. We then counted the number of records returned from the authority file when we used each of the three lookup keys: last name, last name+first initial of first name, and last name+first name. We also counted how often the returned block of authority records contained the matching authority record that had been manually identified. We computed the mean block size by dividing the number of records returned for each key by the number of test instances (i.e., 500). We computed the recall (the probability that block contains matching record) by dividing the number of block sets that contained the manually identified authority record by the number of test instances.

3.6 Selecting Matching Feature Functions

Match features are functions that compare semantically compatible fields from the authority file record and the structured record in such a way that the more closely the argument field values match the higher the feature function value.

The operation of particular feature functions is governed by the semantics of the fields being compared. Two types of similarity functions merit special mention because they work well for many field classes. They are the TFIDF cosine similarity function and the family of string edit distance functions.

The TFIDF cosine similarity function computes the cosine similarity of a string field in the text structure record and a string field value in the authority record. The inverse document frequency values for words in the record field are computed from the field in the authority file. Each record in the authority file is considered a document and the number of different records in which a word appears in the string field is used as the word's occurrence count [4].

The family of string similarity functions includes among others the Hamming distance, Levenshtein distance, Smith-Waterman distance, and Jaro-Winkler dis-

tance. Each of these functions measures the distance between strings as a function of the number of character changes that need to be made to transform one string into another [4].

For our example NE resolution system, we use the following five similarity functions to convert an attorney authority file and a text structured record pair into a feature vector. In our example, we block using last name, so we need no feature function for last name. In our Profiler system, we do incorporate a last name feature function and use a blocking function other than simple last name. For purposes of illustration, however, using last name as the blocking function works adequately.

First name similarity function. This compares the first name of the attorney field in the authority file with the first name in the text structured record. The function returns a value of 1.0 if the names match exactly, a value of 0.8 if the names match as nick names, a value of 0.8 if one name is only specified by an initial and matches the first letter of the other name, a value of 0.5 if one or both of the first names is unspecified, and a value of 0.0 if the names mismatch.

Middle name similarity function. This compares the middle name or initial of the attorney field in the authority file with the middle name or initial in the text structured record. This function is essentially the same as the first name similarity function except it is applied to middle names.

Name suffix similarity function. This feature function compares the name suffix value (e.g., "Jr." or "III") in the authority file record with the name suffix value in the text structured record. This function returns a 1.0 if both suffix are specified and match, a 0.8 if both suffixes are unspecified, a 0.5 if a suffix is specified in one name and not in the other, and a 0.0 if the suffixes are specified and mismatch.

City-state similarity function. This feature compares city-state information in the authority file record with city-state information in the text structured record. The function returns a 1.0 if both city and state match, a 0.75 if just the state matches, a 0.5 if city-state information is not specified in one or both records, and a 0.0 if the states mismatch. Note that we consider all city-state pairs that co-occur in the same paragraph with the attorney name and use the highest scoring city-state pair to set our feature function value.

Firm name similarity function. This feature measures the TFIDF cosine similarity between the law firm name specified in the authority file record and law firm names in the text structured record. Note that we consider all firm names that co-occur in the same paragraph with the attorney name reference and use the highest scoring firm name to set our feature function value.

Note that these feature functions are somewhat simpler than the feature set we use for Profiler. However, these functions work well here to illustrate NE resolution concepts.

3.7 Developing Training Data

To train an NE resolution record pair classifier, you need a relatively large set of authority-text structured record pairs that co-refer (positive data) and a set of

authority-text structure record pairs that do not co-refer (negative data). The positive and negative training data are used by a machine learning program to create a model that combines feature function values to yield a match belief score for a given feature function vector associated with a authority-text structured record pair.

The amount of training data one needs for a NE resolution system varies according to the nature of names, text, and authority file associated with the system. Our experience has been that a few thousand positive and negative record pairs works fairly well.

Training data can be created manually or in some cases automatically. To create training data manually, we follow the same procedure we used to create test data. To create training data automatically, we pick one or two feature fields that under certain conditions can be used to very probably identify a positive match in a candidate record block. We call the feature fields that can automatically identify positive matches surrogate features. Note that negative training data can be generated for NE resolution automatically from positive training instances in the same way they were generated for the test data.

The surrogate feature method of automatically creating positive training data works well if there are non-surrogate features that are independent of the surrogate features and that are robust enough to identify matching records within the candidate authority record block. The attraction of using surrogate features is that we do not have to incur the cost of manually creating training data. The drawback of using surrogate features is that, once we have used them to identify positive matches, we cannot reuse them as match features in our scoring program.

For our example NE resolution system, we created 5,000 manually matched positive training examples.

We also created 70,000 positive instances automatically in our caselaw corpus by using rare first and last name combinations as a surrogate feature. We selected rare names by identifying first and last name pairs in census data that occur fewer than 50 times in the general U.S. population and only once in the attorney authority file. Any attorney file and text file record pairs that have a matching rare first-last name combination are considered positive matches.

3.8 Using Support Vector Machine to Learn Matching Function

A good method of linking structured records to authority records is to use a Support Vector Machine (SVM) classifier that measures the likelihood that a particular structured record and authority record refer to the same entity. An advantage of the SVM is that it can learn non-linear functions of feature values relatively easily.

To create an NE resolution system using an SVM, we provide the SVM with positive and negative training instances to obtain a model of support vectors with which to score feature function vectors. The higher the score the model returns for a feature function the higher the likelihood that the authority file record and text structured record pair refer to the same person.

Table 5. Precision, Recall and F-measure of the NE resolution system using manual and surrogate training data to train SVM matching function.

Training data	Precision	Recall	F-measure
Manual	0.96	0.95	0.95
Surrogate	0.96	0.89	0.92

For our NE resolution example system, we created one matching model from the manual training data and one model from the automatic training data.

3.9 Assembling and Testing the NE resolution Pipeline

We create a NE resolution system by assembling our text structured record generator, blocking function, feature vector generator, and feature vector scorer into a pipeline that will find a matching authority record for a given text name reference. The pipeline functions as follows for a given text reference:

1. Create a text structured record for a text name reference.
2. Create blocking key from the structured record. This value in our tests was the last name of the attorney name tagged in text.
3. Retrieve all records from the authority file that match the blocking key. These are the authority candidate records.
4. Pair each authority candidate record with the current text structure record and compute a feature function vector.
5. Score each feature function vector with the trained SVM.
6. Choose the authority file record associated with the highest scoring feature vector as the best match to the text structure record.
7. If the highest scoring feature vector exceeds a minimum threshold, assign entity id associated with the matching authority file record to the text reference associated with the text structure record.

We measured how well the SVM approach works using manual and automatically acquired training data for the attorney names in caselaw problem. Our results are shown in table 4.

The f-measure using our surrogate training approach was only three percentage points below our manual training approach. The difference was the six percentage point drop in recall. We suspect this drop was due to the fact that some matching attorneys instances involved first names expressed as nick names or initials and our surrogate training examples excluded these types of matches.

Overall however the surrogate approach using rare names was fairly effective. We have discussed using rare names for surrogate training more fully in [5].

4 Conclusion

One aspect of semantic processing of legal texts is figuring out what people, courts, companies, law firms, etc. are mentioned in a text. This processing can

be broken into two parts: (i) recognizing names of certain types and (ii) resolving these names to specific individuals in the world. We have outlined both of these tasks and presented systems for performing them in the legal domain.

References

1. Dozier, C., Haschart, R.: Automatic Extraction and Linking of Person Names in Legal Text. In Proceedings of RIAO 2000 (Recherche dInformation Assistee par Ordinateur), 12-14 April 2000, Paris, France, pp. 1305-1321 (2000)
2. Dozier, C., Zielund, T.:Cross Document Co-Reference Resolution Applications for People in the Legal Domain. In Proceedings of the ACL 2004 Workshop on Reference Resolution and its Applications, 25-26 July 2004, Barcelona, Spain, pp. 9-16 (2004)
3. Chaudhary, M., Dozier, C., Atkinson, G., Berosik, G., , Guo, X., Samler, S.: Mining Legal Text to Create a Litigation History Database. In Proceedings of IASTED International Conference on Law and Technology, Cambridge, MA, USA. (2006)
4. Cohen, W., Ravikumar, P., Fienberg, S.: A Comparison of String Distance Metrics for Name-matching Tasks. Proc. II Web Workshop IJCAI, 2003, pp. 73-78 (2003)
5. Dozier, C., Veeramachaneni, S.: Names, Fame, and Co-Reference Resolution, Thomson Reuters Research and Development Technical Report, 2009.
6. Liao, W., Light, M.: Integrating High Precision Rules with Statistical Sequence Classifiers for Accuracy and Speed. Proc. of the Software engineering, testing, and quality assurance for natural language processing workshop, In Proceedings of NAACL, (2009).
7. Alex Yeh, Alex Morgan, Marc Colosimo, and Lynette Hirschman: BioCreative task 1A: Gene mention finding evaluation., BMC Bioinformatics 6(Suppl. 1) (2005).
8. Ralph Grishman, Beth Sundheim: Message Understanding Conference - 6: A Brief History. In Proceedings of the 16th International Conference on Computational Linguistics (COLING), I, Kopenhagen, (1996).
9. J. Lafferty, A. McCallum, and F. Pereira: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proc. of ICML, 2001.