Litigation Analytics: Extracting and querying motions and orders from US federal courts

Thomas Vacek, Dezhao Song, Hugo Molina-Salgado,
Ronald Teo, Conner Cowling and Frank Schilder
Thomson Reuters R&D
610 Opperman Drive
Eagan, MN 55123, USA
FirstName.LastName@ThomsonReuters.com

Abstract

Legal litigation planning can benefit from statistics collected from past decisions made by judges. Information on the typical duration for a submitted motion, for example, can give valuable clues for developing a successful strategy. Such information is encoded in semi-structured documents called dockets. In order to extract and aggregate this information, we deployed various information extraction and machine learning techniques. The aggregated data can be queried in real time within the Westlaw Edge search engine. In addition to a keyword search for judges, lawyers, law firms, parties and courts, we also implemented a question answering interface that offers targeted questions in order to get to the respective answers quicker.

1 Introduction

Dockets contain valuable meta-data about the legal actions carried out by the parties, the lawyers and law firms representing their clients and the judges presiding over the cases. A detailed record of the activities by the parties involved is kept by the court's clerk and it provides indirect insights into litigation strategies.

The entries of a docket contain the motions filed and orders issued. Understanding those entries unlocks the information that fuels litigation analytics. A detailed manual analysis of a docket could provide valuable information for the suit and the respective judge, but reading through hundreds of dockets would be very time-consuming.

Applying machine learning and NLP capabilities to all federal dockets allowed us to collect this information for 8 million past dockets and also enables us to keep up with all newly closed dockets. The information is being extracted automatically and extractions that are of low confidence identified by the machine are then manually reviewed in order to ensure the quality of the analytics. We extracted about 300,000 parties, 500,000 lawyers, 125,000 law firms and 6,700 judges from 90 million state and federal dockets combined. Approximately 18 million motion and orders were extracted from 8 million federal dockets processed.

This demonstration paper describes in more detail the underlying problems we solved and provides an overview of how we utilized machine learning and manual review in combination with rules (that are designed based on expert knowledge). The remainder of this paper is organized as follows: Section 2 describes previous work followed by Section 3 detailing the way we extracted the information on motions and orders from federal dockets. We briefly describe the annotation study we carried out and how the motions and orders and the chains between them were extracted from the docket. Section 5 shows how users can use natural language questions to directly query judges and how they ruled on various motions (e.g., motion for summary judgment). Section 4 will run through the various steps of the demo showing how a legal researcher would interact with the system and Section 6 concludes.

2 Problem description and previous work

2.1 Motions and orders in the US Federal court

All dockets for the US federal courts are recorded by the PACER (Public Access to Court Records) system. PACER is a US government computer system that provides a front-end to the CM/ECF (Case Management/Electronic Court Filing) system in use in most Federal District Courts. CM/ECF allows counsel, clerks, and judges to quickly access case documents and the documents with the court electronically.

Proceedings of NAACL-HLT 2019: Demonstrations, pages 116–121
Minneapolis, Minnesota, June 2 - June 7, 2019.

A litigation normally starts with the claims filed by the plaintiff and the dockets keep track of the various court actions by the participating parties and the judge. Lawyers may file various motions (e.g., motion to dismiss, motion in limine). The following docket entries show a motion to dismiss in entry 9 and later denied by the judge via an order in entry 25. Note that there are entries in between related to the original motion that could be confused as motions to dismiss.

9 MOTION to dismiss Party Alamo Rent-A-Car

15 REQUEST/STATEMENT of Oral Argument by Shannin Woody regarding [9] [11] motions to Dismiss.

16 RESPONSE/MEMORANDUM in Opposition to motion to dismiss filed by Shannin Woddy

25 ORDER granting [9] Motion to dismiss; granting [11] motion to dismiss

A simple keyword-based approach will fail to reliably extract the motions and orders because the language describing a motion filing is often very similar to filings of replies and sur-replies. In addition, linking the respective orders and motions is not always straight forward because links normally indicated by a number (e.g., [9]) may not be always present. Lastly, the language to indicate motions, although fairly standardized, differs sometimes in language and the details provided by the parties and judges:[1]

- MOTION in Limine by Amber Blackwell, Kevin Blackwell.
- MOTION to Exclude Any Evidence Relating to Recordings of Defendants or Their Agents by Keith Castilla, Uretek USA, Inc., filed. Motion Docket Date 6/7/2013.

2.2 Previous work

The task of automatically parsing docket documents is a relatively novel task and there are only a few approaches that have dealt with information extraction and classification tasks of legal court proceedings. Nallapati and Manning (2008) are one of the few researchers who investigated machine learning approaches applied to classifying summary judgment motions only. Their findings indicated that rule-based approaches showed better results than a machine learning approach such as using a Support Vector machine (SVM). Their results indicated that a classification approach using an SVM achieves only an overall F1-value of about 80, while a specified rule-based approach is able to reach almost 90 F1-value.

More recent work by Branting (2017) addresses the issue of detecting errors in filing motions as well as the matching between motions and orders. They report a mean rank of 0.5-0.6 on this task.

The method for answering questions is described in more detail in previous work (Song et al., 2017, 2015). The capabilities of the parser, however, have been extended in order to answer specific questions on motion rulings and the time to rule. The previous parser version was restricted to case documents and was able to answer questions such as *who many cases has Judge John Tunheim ruled on in 2018.*

3 Motion analysis

We carried out an extensive annotation study for the problem at hand. Multiple domain experts annotated federal dockets with the motion information and detailed relationship information between the docket entries. Similar to Nallapati and Manning (2008), we found machine learning approaches not sufficient and only a combination of rules, ML approaches and editorial review could ensure high quality output.

Data collection Accurate gold data is necessary for hypothesis testing. Given that the editorial definitions have profound implications to the amenability of the task to automation and the significance of the results, we sought an expressive annotation scheme that would capture the language and structure of the text of each docket entry with respect to events that we were interested in. We developed a small set of token classes and structural dependencies for the annotation scheme. Dependencies only exist between these tokens and phrases, and the dependency relations are defined by the meaning of the underlying text, not necessarily its grammatical structure.

Domain experts were instructed to scan each entry of a docket to determine if any part should be annotated. We suspected that domain experts, be-

[1] Both motions are motions in limine, but the second motion does not explicitly use the term of art.

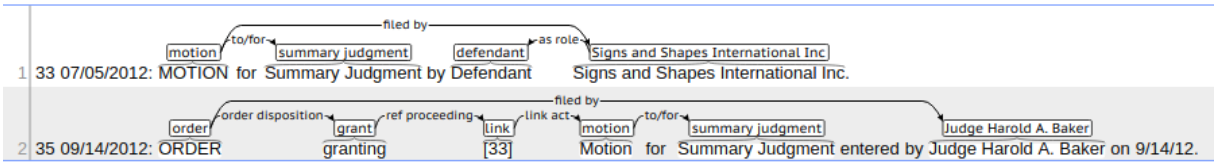

Figure 1: Example of an annotated motion order in the BRAT tool. The annotated tokens were seeded automatically, and domain experts used the drag-and-drop UI to place arcs. Annotations are based on the meaning of the sentence and tied to the most important tokens.

ing human, were susceptible to tunnel vision, i.e. putting a great deal of attention into accurately annotating one docket entry while completely missing another one. Since our token classes are only applicable to tokens related to high-level events in motion practice, more than 75% of docket entries are expected to have no annotations at all. We used the BRAT annotation tool (Stenetorp et al., 2012). Our distinction between token classes and dependencies maps perfectly into BRAT's taxonomy of entities and relations. We found that the token class annotations could be reliably seeded using regular-expression matching rules. The seeded token class annotations amounted to a keyword search, which focused the domain expert's attention on the parts of the document most likely in need of annotation. Seeding the token class annotations removed the most time consuming step of the annotation task, so that the remaining effort almost exclusively involved dragging and dropping relationship arcs. A screenshot of the tool is given at Figure 1.

Docket parsing The core component of the Litigation Analytics system focusses on motion and order detection. The overall system deploys a mix of high-precision and high-recall rules as well as some machine learning models to increase overall recall. First, motions and orders are tagged with high-precision rules. Then motions and orders are parsed in order to extract motion type, filer, order type, decision type, and judge names. Finally, the motions and orders are chained together.

High-precision rules that have very high confidence in identifying the motions in questions extract many of the motions that follow a clearly identifiable pattern. However, such rules may miss unusual language or motion/order description that contain typos or other additional material. In addition to the high-precision rules, we also adopt high-recall rules in order to capture as many motions/orders as possible and a machine learning module to learn the language variants associated with in limine motions, for example. The machine learning component is is a fall-back system for the high-precision rules system and it is triggered when the language variants in the data prevents the high-precision rules from finding a high confidence result. The learner used in this module is a regression SVM (Chang and Lin, 2011) that outputs a score that is fine-tuned for the required precision and recall numbers.

A second system that implements high-recall rules will also derive motions and linked orders. Eventually, the outputs of both components are merged to ensure overall high-precision output. The identified motions/orders are merged and cross-referenced with data we also have manually annotated for a smaller subset of dockets from past products. The merging process focusses on precision, but will also allow for higher recall and additional editorial process.

The output of all the motion analysis component is then ingested into the Litigation Analytics application of Westlaw Edge, demonstrating the analytics by judge, lawyer or law firm. Figure 2 shows how the analytics can be further explored by selecting different views. Users may be interested in specific motions, case types or parties. The app allows the user to explore the entire set of motions extracted from the federal court docket set.

Evaluation The performance of the docket parsing component was evaluated by using the annotated test data. The goal was to achieve high-precision results with acceptable recall. See Table 1 for precision and recall values we achieved. The experiments reported here are not directly comparable with (Nallapati and Manning, 2008), as their task is to detect a court order on summary judgment, whereas our task is to detect the filing of a motion.

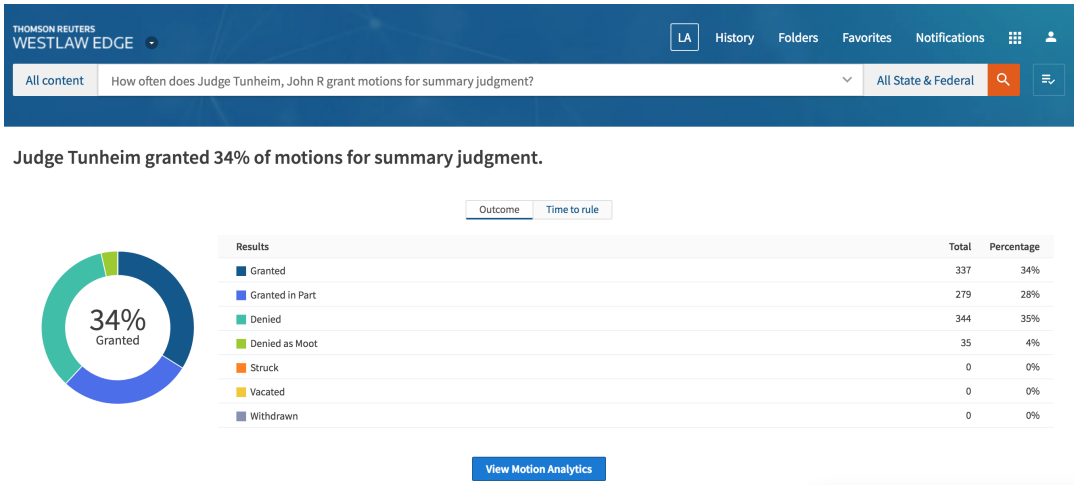

Figure 2: Natural language questions allow direct access to Litigation Analytics feature

Motion	Pecision	Recall
Dismiss	92.4	90.3
Summary Judgment	97.8	92.6
Pro Hac Vice	84.9	93.7
In Limine	94.7	91.5

Table 1: Precision and Recall values for the four most frequent motions

4 Demo

The demo of the Litigation Analytics feature will start with asking a natural language question. Typing in a name judge name into the search box will trigger multiple example questions to be asked.

- How often does Judge Tunheim, John R grant motions for summary judgment?
- How often does Judge Tunheim, John R deny motions for summary judgment?
- How long does it take Judge Tunheim, John R to rule on motions for summary judgment?

The default motion suggested is summary judgment, but the user can also specify other motions such as motion to dismiss or in limine. After selecting one of those questions, the meaning of the questions is computed and an answer in form of a generated sentence as well as a chart is generated (cf. Figure 2).

The user can link to the Litigation Analytics tool by clicking on the View Motion Analytics button. A more comprehensive view of the data collected for the respective judge will be displayed and the user can analyze the data and plan their litigation strategy. A bar chart, for example, will provide an overview of up to 23 different motions and the collected analytics. The user may drill down to another motion and is able to click through the actual docket for a particular motions or decision if desired. The overview page will also show a direct comparison between the judge's time to rule and their peers on the respective court (cf. 3).

5 Natural Language Interface

In order to enable our customers to easily find the exact information they are looking for, we developed a natural language interface *TR Discover*. Given a natural language question, we first parse it into its First Order Logic representation (FOL) via a feature-based context free grammar (FCFG). The grammar defines the options available to the user and implements the mapping from English into logic. A second translation step then maps from the FOL representation into a standard query language (e.g., a SQL or a Boolean query), allowing the translated query to rely on robust existing technology. Since all professionals can use natural language, we retain the accessibility advantages of keyword search, and since the mapping from the logical formalism to the query language is information-preserving, we retain the precision of query-based information access.

Our FCFG consists of phrase structure rules (i.e., grammar rules) and lexical entries (i.e., lexicon). The majority of our grammar rules are domain independent allowing the grammar to be portable to new domains (e.g., Tax). *G1* and *G2*

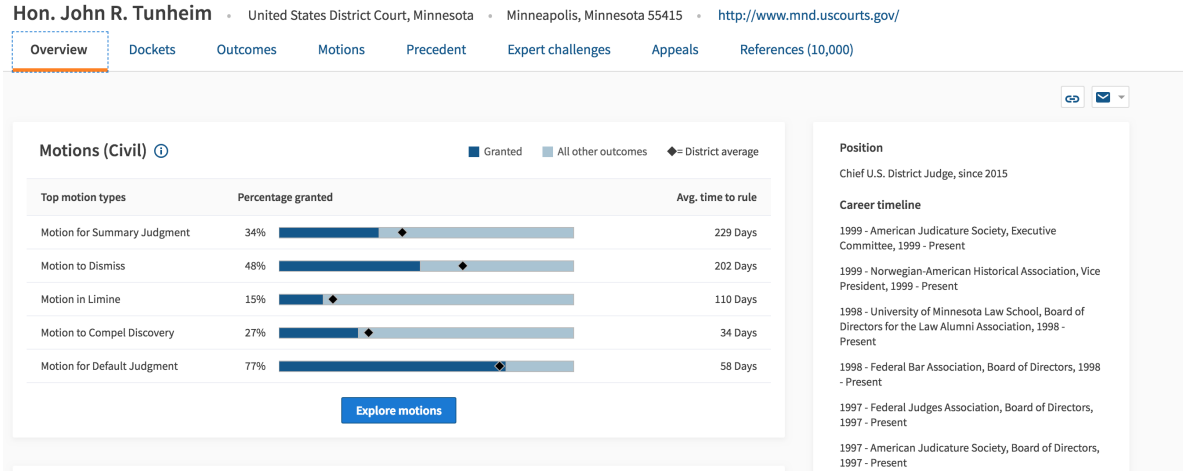

Figure 3: Average time to rule allows comparison with other judges

are two example grammar rules. Specifically, Rule *G2* indicates that a verb phrase (*VP*) contains a verb (*V*) and a noun phrase (*NP*).

- G1: NP → N
- G2: VP → V NP

Furthermore, each lexical entry in the FCFG contains a variety of domain-specific features that are used to constrain the number of parses computed by the parser preferably to a single, unambiguous parse. *L1* and *L2* are two examples of our lexical entries.

- L1: N[TYPE=motion, NUM=sg, SEM=<λx.mtn(x)>] → 'motion'
- L2: TV[TYPE=[judge,motion,grant], SEM=<$\lambda X\ x.X(\lambda y$.grant_judge_mtn(x,y))>, TNS=prog, NUM=?n, -PASS] → 'grant'

Here, *L1* is the lexical entry for the term *motion*, indicating that it is of TYPE *motion*, is single ("NUM=sg"), and has the semantic representation $\lambda x.mtn(x)$. Verbs (V) have additional features, such as tense (TNS) and TYPE, as shown in *L2*. The TYPE of verbs specify both the potential subject-TYPE and object-TYPE. A general form for specifying the subject and object types for verbs is as following: TYPE=[subject_constraint, object_constraint, predicate_name]. With such type constraints, we can then license the question *motion for summary judgment granted by Judge John R. Tunheim* while rejecting other questions like *motion for summary judgment granted by Attorney John R. Tunheim* on the basis of the mismatch in semantic type.

By using our FCFG, the question *How often does Judge Tunheim, John R grant motions for summary judgment?* can then be parsed into the following FOL representation:

$$count(user, P1))\wedge$$
$$\forall P1.((((label(P1, summary\_judgment))\wedge$$
$$(type(P1, motion))) \rightarrow$$
$$((\exists P2.(((grantedBy\_motion\_judge(P1, P2))$$
$$\wedge(label(P2, Tunheim, John, R, \_))\wedge$$
$$(type(P2, judge)))))))$$

This FOL representation is further translated into a SQL or Boolean query in order to retrieve the search results.

6 Conclusions

This demo shows how various machine learning and NLP techniques can be used to (a) get access to analytical data more quickly via a natural language interface and (b) to create data from semi-structured documents such as legal dockets. The Westlaw Edge product provides legal researchers access to this newly created analytics for judges and courts in order to formulate their litigation strategy. Motion Analytics has unlocked a large trove of data that up until now required painstaking manual review to glean useful insights. This product gives attorneys, clients, and the public a new level of insight into the litigation process.

Future research will focus on exploring machine learning methods and transfer learning techniques in order to apply our findings to other jurisdictions as well.

References

Luther Karl Branting. 2017. Automating judicial document analysis. In *Proceedings of the Second Workshop on Automated Semantic Analysis of Information in Legal Texts co-located with the 16th International Conference on Artificial Intelligence and Law (ICAIL 2017), London, UK, June 16, 2017.*

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Ramesh Nallapati and Christopher D Manning. 2008. Legal docket-entry classification: Where machine learning stumbles. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 438–446, Honolulu, Hawaii. Association for Computational Linguistics, Association for Computational Linguistics.

Dezhao Song, Frank Schilder, Shai Hertz, Giuseppe Saltini, Charese Smiley, Phani Nivarthi, Oren Hazai, Dudi Landau, Mike Zaharkin, Tom Zielund, et al. 2017. Building and querying an enterprise knowledge graph. *IEEE Transactions on Services Computing*.

Dezhao Song, Frank Schilder, Charese Smiley, Chris Brew, Tom Zielund, Hiroko Bretz, Robert Martin, Chris Dale, John Duprey, Tim Miller, and Johanna Harrison. 2015. TR Discover: A Natural Language Interface for Querying and Analyzing Interlinked Datasets. In *The Semantic Web - ISWC 2015*, volume 9367 of *Lecture Notes in Computer Science*, pages 21–37. Springer International Publishing.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: A web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 102–107, Stroudsburg, PA, USA. Association for Computational Linguistics.