Extracting Contract Elements

Ilias Chalkidis
Ion Androutsopoulos
Department of Informatics
Athens University of Economics and Business
Athens, Greece

Achilleas Michos
Cognitiv+
London, UK

ABSTRACT

We study how contract element extraction can be automated. We provide a labeled dataset with gold contract element annotations, along with an unlabeled dataset of contracts that can be used to pre-train word embeddings. Both datasets are provided in an encoded form to bypass privacy issues. We describe and experimentally compare several contract element extraction methods that use manually written rules and linear classifiers (logistic regression, SVMs) with hand-crafted features, word embeddings, and part-of-speech tag embeddings. The best results are obtained by a hybrid method that combines machine learning (with hand-crafted features and embeddings) and manually written post-processing rules.

CCS CONCEPTS

•Computing methodologies → Information extraction; •Applied computing → Law;

KEYWORDS

Natural language processing, machine learning, legal text analytics, information extraction, contracts, datasets, evaluation.

ACM Reference format:
Ilias Chalkidis, Ion Androutsopoulos, and Achilleas Michos. 2017. Extracting Contract Elements. In *Proceedings of International Conference on Artificial Intelligence and Law, London, UK, June 12–15, 2017 (ICAIL'17),* 10 pages.
DOI: http://dx.doi.org/10.1145/3086512.3086515

1 INTRODUCTION

Contracts are legal texts describing agreements. Law firms, companies, government agencies etc. need to monitor contracts for a wide range of tasks [24]. For example, law firms need to notify their clients when contracts are about to expire or when contracts are affected by legislation amendments. Large contractors need to keep track of agreed payments. Taxation authorities may need to focus on contracts involving particular parties and large payments. Many of these tasks can be automated by extracting particular contract elements (e.g., termination dates, legislation references, contracting parties, agreed payments). Contract element extraction, however, is currently performed mostly manually, which is tedious and costly.

In this paper, we study how contract element extraction can be automated. As a starting point, we provide a new benchmark dataset of approximately 3,500 English contracts, manually annotated with 11 types of contract elements. This dataset can be used to train and test contract element extraction algorithms. To bypass privacy issues, the benchmark dataset is provided in an encoded form, where each vocabulary word has been replaced by a unique integer identifier, as in some spam filtering datasets [1]. For example, each occurrence of the word 'termination' may have been replaced by the integer identifier '3156'. We also provide hand-crafted features per token (word occurrence) of the benchmark dataset, for example indicating the part-of-speech (POS) tag and length (in characters) that each token had before it was replaced by an integer identifier, whether the (original) token contained numerical characters etc.

Furthermore, we provide word embeddings [14–16] per vocabulary word (integer identifier) of the benchmark dataset. Roughly speaking, word embeddings are dense real-valued vectors, each representing a particular vocabulary word as a point in a high-dimensional vector space, such that the vectors of words with similar morpho-syntactic and/or semantic properties are close in the high-dimensional space. Word embeddings have led to significant improvements in several natural language processing tasks in recent years, and can be produced (pre-trained) in an unsupervised manner from large unlabeled corpora [21, 23, 26]. The word embeddings that we provide were obtained (pre-trained) by applying WORD2VEC [22] to an additional unlabeled dataset of approximately 750,000 contracts, which we also make available in the same encoded form. We also provide POS tag embeddings, which were obtained by applying WORD2VEC to contracts from the unlabeled dataset, after replacing the words by their POS tags.[1]

Using the above datasets, we experiment with both manually written contract element extraction rules and trainable linear classifiers, namely Logistic Regression (LR) [19, 30] and linear Support Vector Machines (SVMs) [4, 29]. When experimenting with LR and SVMs, we use hand-crafted features, pre-trained word embeddings, and/or pre-trained POS tag embeddings. The best results are obtained by a hybrid method that combines machine learning (LR or SVM, with hand-crafted features, word and POS tag embeddings) and manually written post-processing rules. The F_1-score of the hybrid method exceeds 0.8 for all but one of the contract element types of our benchmark dataset. We view the methods of this paper as strong baselines for further work that may experiment with more complex classifiers (e.g., convolutional or recurrent neural networks [9, 10]) using the datasets we provide.[2]

Overall, the main contributions of this paper are the following:

ICAIL'17, London, UK

DOI: http://dx.doi.org/10.1145/3086512.3086515

[1] We use NLTK's (v. 3.2.1) default POS tagger (http://nltk.org/).
[2] The datasets will be available from http://nlp.cs.aueb.gr/.

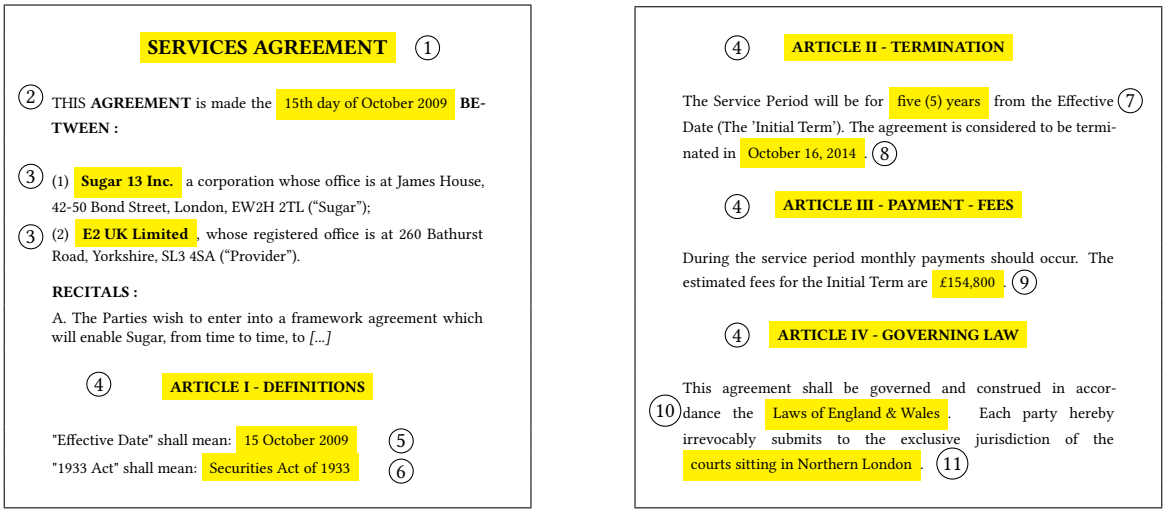

Figure 1: Typical structure of a contract, with contract elements highlighted.

1. The paper studies a legal text analytics task, contract element extraction, which has significant practical value (Section 2). This task has not been studied at such fine granularity in previous work (e.g., some previous work classifies entire contract lines, sentences, or clauses, rather than identifying contract elements) and/or it has been studied using much smaller datasets, fewer types of contract elements, or without considering embeddings (Section 6).

2. The paper describes and is accompanied by a new benchmark dataset of approximately 3,500 English contracts with gold contract element annotations (Section 3), which can be used to train and test contract element extraction algorithms. A larger unlabeled dataset of approximately 750,000 English contracts is also provided and can be used to pre-train word and POS tag embeddings (e.g., with other algorithms than the one we used). Pre-trained word and POS tag embeddings, as well as hand-crafted features, are also provided for the tokens of both datasets. An encoding that replaces words by unique integer identifiers is also adopted to bypass privacy issues, and may also prove useful in other legal text analytics tasks.

3. Several contract element extraction methods are presented (Section 4), including machine learning methods (LR, SVMs) with hand-crafted features and/or word and POS tag embeddings, as well as manually written rules (used instead of the machine learning classifiers or to post-process their decisions). Experimental results (Section 5) show that some of the methods are particularly strong baselines for future work, with the best results (F_1-score exceeding 0.8 in all but one contract element type) achieved by a hybrid method that combines machine learning (using all of the features considered) and manually written post-processing rules.

2 THE TASK

This section defines the contract element extraction task addressed by this paper. It describes the typical structure of a contract, the contract elements that we aim to extract, the zones (parts) of the contracts the elements are extracted from, also highlighting possible applications of contract element extraction and differences from generic named entity recognition [3, 25].

2.1 Contract Structure and Elements

Contracts typically start with a *preamble*, which contains the contract title (Fig. 1, point 1) and specifies the start or effective date (point 2) and contracting parties (points 3). It is also common to use a *cover page* with the same information followed by a *table of contents*, before the preamble. The preamble is usually followed by the *recitals* (Fig. 1), which provide background information. The remainder of the contract is organized in *clauses*, often called 'chapters', 'articles', 'sections' etc. to reflect a hierarchical structure. Clauses have headings (Fig. 1, points 4), which indicate their topics (e.g., 'Definitions', 'Termination', 'Payments').

In this paper, we focus on extracting the following types of *contract elements*, when present. More contract element types can be defined, but the types we focus on are common and particularly useful in analytics applications like the ones we highlight below.

Contract Title (Fig. 1, point 1). The title usually indicates the type of the contract (e.g., services, employment, loan) and often also the version of the contract (e.g., 'second amendment'). Hence, extracting contract titles is a useful step towards classifying contracts per type and creating threads with multiple versions of the same contract (e.g., clustering contracts of the same type that involve the same contracting parties, and ordering them by version).

Contracting Parties (Fig. 1, points 3). Extracting contracting parties allows building inverted indices to quickly retrieve contracts that involve particular parties, which is a common type of query. By extracting contracting parties one can also build graphs showing the interdependencies between companies, contractors etc.

Start, Effective, Termination Dates, Contract Period, Value: The *start date* (Fig. 1, point 2) is when the contract was signed. The *effective* and *termination dates* (points 5, 8) specify when the contract becomes effective and terminates, respectively. The *contract period* (point 7) is the number of working or calendar days the contract will be effective for. The *contract value* (point 9) is the price of the agreed transaction (e.g., salary, lump sum). These elements are particularly useful when searching for contracts that were active

Extraction Zones (at testing)	Example Clause Heading Words	Contract Elements Typically Included
Cover page and preamble	–	Contract Title, Contracting Parties, Start Date, Effective Date
Term clause	'Term', 'Period', 'Term of Agreement'	Termination Date, Contract Period
Termination clause	'Termination', 'Termination of Agreement'	Termination Date
Governing Law clause	'Governing Law', 'Applicable Law'	Governing Law, Jurisdiction
Jurisdiction clause	'Jurisdiction', 'Jury Trial', 'Venue'	Jurisdiction
Miscellaneous clause	'Miscellaneous', 'Entire Agreement'	Governing Law, Jurisdiction
Contract Value clause	'Lump Sum', 'Salary'	Contract Value
In the text after the recitals, zones starting up to 20 tokens before and ending up to 20 tokens after each line break, not crossing other line breaks		Clause Headings
In the entire contract, zones starting up to 20 tokens before and ending up to 20 tokens after each occurrence of words like 'Act', 'Treaty' etc.		Legislation References

Table 1: Extraction zones where contract elements of different types are searched during testing.

at particular times or that have particular durations or values, for example to detect suspicious transactions, possibly in combination with searching for contracts involving particular contracting parties. They can also be used to notify clients when contracts they are involved in are about to become effective or terminate.

Governing Law, Jurisdiction, Legislation Refs: The *governing law* (Fig. 1, point 10) specifies the country or state whose laws apply. The *jurisdiction* (point 11) specifies the courts responsible to resolve disputes. The *legislation references* (point 6) are laws the contract depends on. These elements are useful, for example, when contracts need to be routed to experts on the laws of particular states or countries and/or when contracts need to be revised following legislation revisions (e.g., when laws are amended or replaced).

Clause Headings (Fig. 1, points 4) are needed, for example, to automatically build a table of contents (for contracts that do not include one) and split the text of each contract after the recitals into clauses (from clause heading to clause heading). We also note that most clause headings contain typical words or phrases that indicate their topics. For example, many contracts include a clause whose heading contains phrases like 'Governing Law' or 'Applicable Law'; the governing law contract element (e.g., 'Laws of England & Wales' in Fig. 1, point 10) is then found in that clause. Similarly, many contracts include a clause whose heading contains words or phrases like 'Termination' or 'Termination of Agreement'; the termination date (e.g., 'October 16, 2014' in Fig. 1, point 8) is then found in that clause. Alternatively, other contracts include a more general clause whose heading contains words or phrases like 'Term', 'Period', 'Term of Agreement'; clauses of this kind typically include the termination date and/or the contract period. Hence, identifying clause headings (and then looking for indicative words or phrases in the headings) is also a first step towards identifying clauses where other contract elements are expected to be found.

In fact, each type of contract element is almost always found in particular types of clauses or other zones (e.g., preamble, cover page) of the contracts, hereafter jointly called *extraction zones*. For example, the contracting parties can always be found in the cover page (if present) and preamble. Looking for contracting parties in other parts of the contracts would unnecessarily increase the time needed to process the contracts. In practice, it would also increase the false positives (tokens wrongly identified as contracting parties) during testing, without any other benefit. More importantly, it would also greatly increase the number of negative instances during training (examples of tokens that should not be identified as contracting parties) leading to severe class imbalance (many more negative than positive training instances). This might mislead machine learning algorithms to learn to classify all instances in the majority (negative) class, i.e., never classify tokens as contracting parties. Similar comments can be made for all the other contract element types. Table 1 summarizes the extraction zones where contract elements of different types can be expected to be found.[3]

Hence, when using contract element extraction methods in deployed systems, we first identify the cover page, the preamble, and the zone after the recitals of each contract; this can be easily achieved using simple regular expressions, which work very reliably in practice. We then apply the method(s) that identify clause headings (discussed in Section 4 below) to the zone after the recitals of each contract; these methods are also reasonably reliable (the F_1-score of the best clause heading extraction method is 0.89). Subsequently, we split the text after the recitals into headings (from clause heading to clause heading), and we identify the topic of each clause (term clause, termination clause, governing law clause etc., lines 3–8 of Table 1) using manually crafted lists of indicative words or phrases (column 2 of Table 1), which also work very reliably in practice.[4] The methods that extract the other types of contract elements (also discussed in Section 4 below) are then applied only to the corresponding extraction zones of Table 1.

The *test* part of the labeled dataset that we provide (discussed in Section 3 below) includes annotations that indicate the gold (correct) spans of the extraction zones per contract element type of each test contract (as in Table 1).[5] This allows other researchers to directly compare the results of their core contract element extraction algorithms against our results (presented in Section 5), assuming that the extraction zones have been identified without errors. To reduce the manual annotation effort that was required to produce the benchmark dataset, the process that we use to construct *training* instances for our contract element extraction methods is

[3] The list of words that trigger legislation reference extraction zones (e.g., 'Act', 'Treaty', last line of Table 1) was constructed by inspecting only the training part of the labeled dataset and is included in the files of the datasets we provide.

[4] The lists of indicative words that we use to detect the topics of the clauses were constructed by inspecting only the training part of the labeled dataset. The lists are included in the files that will accompany the datasets we provide.

[5] The extraction zones of clause headings and legislation references are not actually annotated in the test contracts, but they can be trivially reconstructed using instructions provided in the files that accompany the datasets.

slightly different (discussed in Section 3 below) and does not require knowing the correct extraction zones of the training contracts.

2.2 Relation to Named Entity Recognition

Generic named entity recognizers (NERs) [3, 25], which typically recognize persons, organizations, locations, dates, amounts, etc., are not directly applicable to contract element extraction without retraining them on contracts and possibly modifying their feature sets.[6] For example, a generic NER may recognize dates, but without distinguishing between start, effective, termination and other dates (e.g., payment or delivery dates, which we do not aim to extract). Note that several of these date types may occur in the same extraction zones; for example, start and effective dates typically occur both in the cover page and the preamble. Hence, the date types we aim to extract cannot be distinguished simply by observing the extraction zones they occur in. Similarly, a generic NER may recognize amounts without distinguishing between contract values and other amounts (e.g., monthly payments, collateral fees), which may be present in the same extraction zones. It may also recognize persons and organizations, but not all persons and organizations mentioned in a contract are contracting parties; for example, a law firm that prepared the contract or a third-party service provider may be mentioned (sometimes in the same extraction zones as the contracting parties), without being contracting parties. Similar comments apply to the governing law and jurisdiction elements, which are not simply locations. Furthermore, contract titles, clause headings, legislation references are not supported by generic NERs.

Another complication is that contracts often contain abbreviations (e.g., '(Sugar)', '(Provider)' in Fig. 1, points 3), which we do not wish to extract as contracting parties (we wish to extract only 'Sugar 13 Inc.' and 'E2 UK Limited' as contracting parties in Fig. 1). The same applies to abbreviations of legislation references (e.g., '1933 Act' is an abbreviation of 'Securities Act of 1933' in Fig. 1, point 3). Nevertheless, future work (Section 7) could retrain generic NERs on the training part of our annotated benchmark dataset (Section 3), possibly after modifying their features, to recognize the contract element types of Section 2.1 and cope with the challenges highlighted above, comparing against our methods (Section 4).

3 DATASETS

This section describes the encoded datasets that we provide, as well as the hand-crafted features, word embeddings, and POS tag embeddings that we provide for the tokens of the datasets.

3.1 Labeled Benchmark Dataset

The labeled benchmark dataset contains 993 contracts (893 training, 100 test) annotated with gold (correct) clause headings (Fig. 1, points 4), and 2461 contracts (2,111 training, 350 test) with gold annotations for the other 10 types of contract elements (Section 2.1). Table 2 shows the number of contract elements (instances) and tokens per contract element type in the labeled dataset (jointly for the training and test part of the dataset); a contract element may consist of multiple tokens, which is why we also report the number of tokens.[7] The dataset contains approximately 37.1 million tokens (word occurrences) in total.

Contract Element Type	Instances	Tokens
Contract Title	4,363	17,282
Contracting Parties	8,235	34,560
Start Date	2,519	10,990
Effective Date	734	3,218
Termination Date	534	2,140
Contract Period	421	1,628
Contract Value	1062	2,714
Governing Law	2,956	15,497
Jurisdiction	1,841	13,095
Legislation Refs	5,997	32,472
Clause Headings	38,269	~183K

Table 2: Statistics of the labeled dataset.

The gold contract element annotations of the labeled dataset were provided by 10 law students. Each contract was annotated by one student. Before the final annotation, however, we used three rounds of preliminary experiments and two pairs of annotators (the same pairs in all rounds) to measure inter-annotator agreement and improve the annotation guidelines. In each round, 5 new training contracts were given to each pair (10 contracts per round in total) and agreement was measured as $\frac{|A \cap B|}{\max(|A|, |B|)}$, where A, B are the sets of contract elements marked by the two annotators, respectively. The average agreement was approximately 0.27 in the first round, but reached 0.80 in the third round as the guidelines were improved.

All the contracts of the labeled dataset are in English. We cannot reveal their actual texts, due to privacy and IPR issues, but we provide them in an encoded form, where each vocabulary word has been replaced by a unique integer, as already discussed. We also provide hand-crafted features, word embeddings, and POS tag embeddings per token, further discussed below. This arrangement allows experimenting with alternative contract element extraction methods that may rely on bags of words (bags of integer identifiers), the provided hand-crafted features, the provided word and/or POS tag embeddings, or other embeddings (e.g., of different dimensionalities or produced using other algorithms than WORD2VEC) that can be pre-trained on the unlabeled dataset we provide (also discussed below). The encoding, however, does not allow considering the characters or other (than the ones we provide) hand-crafted features of the tokens. Also, it does not allow the actual texts to be studied, though we can provide confidential samples.

The *test* contracts of the labeled dataset also include gold (correct) annotations of the *extraction zones* per contract element type, as already discussed (Section 2.1), i.e., at test time the contract element extraction methods look for contract elements only in the corresponding gold extraction zones (Table 1). For each test contract and each contract element type (e.g., contracting parties), the tokens (word occurrences) of the corresponding extraction zones that are parts of contract elements of that type (e.g., the tokens of contracting parties, as indicated by the gold contract element annotations) are treated as *positive* test instances (e.g., tokens that

[6] Consider, for example, the NERs of Stanford University (http://nlp.stanford.edu/software/CRF-NER.shtml) and spaCy (http://spacy.io/docs/usage/entity-recognition).

[7] We use NLTK's (v. 3.2.1) default tokenizer and sentence splitter (http://nltk.org/).

should be classified as contracting parties), whereas the other tokens of the extraction zones are treated as *negative* test instances (e.g., tokens that should not be classified as contracting parties). To reduce the manual annotation effort that was required to produce the labeled dataset, the *training* contracts of the dataset do not contain gold annotations of the extraction zones. Instead, we train the contract element extraction methods on automatically generated *pseudo-extraction zones* of the training contracts. For each contract element type (e.g., contracting parties), the pseudo-extraction zones of a training contract contain the tokens (word occurrences) of the contract elements of that type (e.g., the tokens of the contracting parties, as indicated by the gold contract element annotations of the training contract) and up to 50 tokens before and after each contract element of that type in the training contract (e.g., 50 tokens before and after each contracting party). The tokens of the contract elements of the particular type are treated as *positive* training instances (e.g., tokens that should be classified as contracting parties), whereas the other (surrounding) tokens of the pseudo-extraction zones are treated as *negative* training instances (e.g., tokens that should not be classified as contracting parties).

In the case of methods that extract start dates, we also add to the negative training instances (training tokens that should not be classified as start dates) all the training tokens (both positive and negative) of the pseudo-extraction zones of the effective dates (excluding tokens that are included in the positive training instances of start dates), and similarly for effective dates, i.e., we add all the training tokens (both positive and negative) of the pseudo-extraction zones of the start dates to the negative training tokens of the effective dates. This helps our (trainable) methods learn to distinguish start from effective dates, which occur in the same parts of the contracts (cover page and preamble). The same arrangement applies to the training instances (training tokens) of governing law and jurisdiction contract elements, which also often occur in the same parts of the contracts. Similarly, in the case of methods that extract legislation references, we add to the negative training tokens all the occurrences of words like 'Act', 'Treaty' etc. (we use the same list of words that trigger extraction zones at test time, last line of Table 1) that have not been annotated as positive training tokens and up to 50 tokens before and after those word occurrences (excluding surrounding tokens that are positive training instances). This helps our methods learn to distinguish legislation references (e.g., 'Securities Act of 1993', Figure 1, point 5) from abbreviations of legislation references (e.g., '1993 Act' in Figure 1).

Especially for clause headings, the extraction zones are the same during both testing (Table 1) and training (no separate rules for pseudo-extraction training zones are needed). They are zones entirely located in the text after the recitals of each contract, with each zone starting up to 20 tokens before and ending up to 20 tokens after a line break, without crossing other line breaks.

3.2 Hand-Crafted Features and POS Tags

For each token (word occurrence) of the labeled benchmark dataset and each contract element type (e.g., contracting parties), we also provide the values of hand-crafted features (17 to 21 features, depending on the type of contract elements being detected). The values of these features are automatically computed; by 'hand-crafted' we mean that the particular feature sets and the meaning of each feature (what each feature stands for) were chosen by ourselves; by contrast the components of word embedding vectors cannot be directly mapped to human-interpretable concepts. The first 14 hand-crafted features are the same regardless of the type of contract element being detected: 4 binary features for all upper, all lower, mixed case tokens, tokens containing numbers; 7 binary features indicating the length of the token (the first feature is true if the length of the token is 1-2 characters, the second feature is true if the length is 3-4 characters, and similarly for lengths 5-6, 7-8, 9-10, 11-12, >12); 3 binary features indicating if the token is numeric, a special character, or stop-word. The other 3-7 features are also binary, but differ per contract element type. They indicate if the token is common inside or near elements of the particular type, or if it is matched by regular expressions that detect frequent parts of elements of the particular type (e.g., '1.1', '1.2.1' for clause headings, '2009', '13th' for start, effective, termination dates).

We also provide the POS tag of each token (word occurrence) of the labeled dataset, as predicted by a generic POS tagger.[8] The tagger uses 45 distinct POS tags. In experiments that employ POS tags directly (not POS tag embeddings, discussed below), we use 45 binary features to indicate the POS tag of each token. Hence, the total number of hand-crafted features becomes 62-66 in this case. The hand-crafted features and POS tags are provided only for the tokens of the labeled datasets, not for the tokens of the unlabeled dataset (discussed next).

3.3 Unlabeled Dataset and Embeddings

As already noted, word embeddings are dense real-valued vectors, each representing a particular vocabulary word as a point in a high-dimensional vector space, such that vectors of words with similar morpho-syntactic and/or semantic properties will be close in the high-dimensional space [14–16]. Word embeddings have led to significant improvements in several natural language processing tasks in recent years, and can be produced (pre-trained) in an unsupervised manner from large unlabeled corpora, for example using tools like WORD2VEC [21, 23] and GLOVE [26]. We applied WORD2VEC (skip-gram model) [22] to an unlabeled dataset of approximately 750,000 English contracts (approx. 9 billion tokens), after encoding the unlabeled dataset in the same way as the labeled one, i.e., each vocabulary word was replaced by an integer identifier. We produced 200-dimensional word embeddings from the unlabeled dataset, one for each integer identifier of a vocabulary word.[9]

The unlabeled dataset, which we also make publicly available in its encoded form, does not contain gold annotations of contract elements and extraction zones; hand-crafted features and POS tags are also not provided. To generalize across numbers with similar patterns and tokens that differ only in the use of upper and lower case, the unlabeled dataset was pre-processed to lower-case its tokens and replace all digits by 'D'. For example,

[8] We use NLTK's (v. 3.2.1) default POS tagger (http://nltk.org/).

[9] We used Gensim's (v. 0.12.4) implementation of WORD2VEC (http://radimrehurek.com/gensim/), with 10 minimum occurrences per word and default values for other parameters. The vocabulary size of WORD2VEC was 514,369. Out of vocabulary words are mapped to random embeddings. Dimensionalities from 100 to 600 are common in natural language processing tasks, with 200-300 dimensions being particularly common. Larger dimensionalities slow down the experiments, often without significant benefit.

Word	10 Closest Words (cosine similarity of embeddings)
'agreement'	'this', 'agreements', 'the', 'herein', 'and', 'hereof', ')', 'terms', 'D', 'company'
'november'	'february', 'august', 'april', 'july', 'june', 'october', 'march', 'january', 'september', 'december'
'inc.'	'inc..', 'corp.', 'inc', 'llc', 'llc.', 'ltd.', 'f/k/a', 'd/b/a', 'l.p.', 'l.l.c.'
'laws'	'law', 'state', 'jurisdictions', 'statutes', 'legislation', 'laws', 'sky', 'province', 'blue', 'authorities'
'court'	'competent', 'tribunal', 'courts', 'sitting', 'arbitrator', 'chancery', 'arbitral', 'judge', 'quasi-judicial', 'proceeding'
'act'	'exchange', 'regulations', 'rules', 'sarbanes-oxley', 'seq.', 'DDd-DD', 'promulgated', 'oxley', 'DDDD', 'gramm-leach-bliley'
'article'	'section', 'D.D.D', 'paragraph', 'DD.D', 'sections', 'D.D', '-', 'subsection', 'clause', 'definitions'

Table 3: Some words that are common in contract elements (left column) and their closest words (right column) in terms of cosine similarity of word embeddings, using the 200-dimensional word embeddings produced from the labeled dataset.

'Agreement' became 'agreement', 'October 16, 2014' became 'october DD, DDDD', 'november 21, 2016' became 'november DD, DDDD'. This pre-processing took place before the encoding of the unlabeled dataset and the subsequent application of WORD2VEC. Consequently, WORD2VEC produced word embeddings for (the integer identifiers of) 'agreement' and 'DDDD', but not 'Agreement' and '2014'. By contrast, the tokens of the labeled dataset were not pre-processed (e.g., there are different integer identifiers for 'Agreement', 'agreement', '2014', '2016'). We provide, however, a mapping from the integer identifiers of the labeled dataset to the integer identifiers of the unlabeled dataset (e.g., in effect showing that the tokens '2014' and '2016' of the labeled dataset both correspond to the token 'DDDD' of the unlabeled dataset), which allows one to link the tokens of the labeled dataset to word embeddings obtained from the unlabeled dataset.

Note that tools like WORD2VEC and GLOVE produce word embeddings by examining only the co-occurrences of the words in an unlabeled corpus, not the characters of the words. Hence, alternative word embeddings (e.g., with different dimensionalities or produced by GLOVE instead of WORD2VEC) can also be generated from the (encoded) unlabeled dataset that we provide, though other methods that produce word embeddings by considering also the characters (or morphemes) of the words [17, 18] cannot be used.

We also experimented with generic pre-trained word embeddings (e.g., obtained from Wikipedia), but the experimental results were much worse, possibly because legal expressions are underrepresented in generic corpora.[10] Also, some collections of generic embeddings do not provide embeddings for stop-words and numbers. For example, there would be no embeddings for '23' and '2013' in 'October 23, 2013', whereas we use the embeddings of 'DD' and 'DDDD' instead. For illustrative purposes, Table 3 shows some words (or tokens) that are frequent in contract elements (e.g., 'agreement') and their 10 closest words, using cosine similarity of word embeddings as the distance measure, and the 200-dimensional word embeddings produced from the unlabeled dataset.

We also provide 25-dimensional POS tag embeddings, which were obtained by applying WORD2VEC (again, skip-gram model, same other settings) to 49,777 contracts from the unlabeled dataset, after replacing the words by their POS tags, again as predicted by a generic POS tagger. We use fewer dimensions in the POS tag embeddings compared to the word embeddings (25 instead of 200), because the POS tag embeddings need to represent only 45 points (POS tags) in their vector space, whereas the word embeddings need to represent the entire vocabulary.

[10] For generic embeddings see, for example, http://nlp.stanford.edu/projects/glove.

4 ELEMENT EXTRACTION METHODS

This section describes the contract element extraction methods that we developed and experimented with. They include machine learning-based classifiers operating on sliding windows of the contracts, as well as manually written rules that replace the machine learning classifiers or post-process their decisions.

4.1 Sliding Window Classifiers

The first contract element extraction method, named **SW-LR-EMB**, uses a separate Logistic Regression (LR) classifier [19, 30] per contract element type (11 classifiers).[11] During testing, each LR classifier scans the tokens of the corresponding extraction zones (Table 1) and classifies each token as positive (part of a contract element of the corresponding type) or negative (not part of a contract element of the corresponding type). For each token t being classified, each classifier considers a sliding window of 5-6 tokens around t (11-13 tokens); the exact size of the window varies, depending on the type of contract elements that each classifier extracts. The window is turned into a feature vector containing the concatenated embeddings of the window's tokens (e.g., 11 tokens $\times$ 200 dimensions = 2,200 features). At training time, each one of the 11 LR classifiers is trained on the positive and negative tokens of the corresponding pseudo-extraction zones (Section 3.1), again using sliding windows of 11-13 tokens. The second method, **SW-SVM-EMB**, is identical, except that it uses 11 linear SVMs [4, 29] instead of LR classifiers.

The next two methods, named **SW-LR-HCF** and **SW-SVM-HCF**, again use a separate classifier (LR or linear SVM, respectively) per contract element type, and sliding windows of 11-13 tokens, but the feature vector of each window now contains the 17-21 hand-crafted features of each token in the window (e.g., 11 tokens $\times$ 17 = 187 features) and the additional 45 binary features per token that indicate the POS tag of each token in the window (e.g., 11 tokens $\times$ 45 = 495 features), instead of the concatenated word embeddings.

A fifth method, named **SW-LR-ALL**, is the same as SW-LR-EMB and SW-LR-HCF, except that the feature vector of each window now contains the concatenated word embeddings, POS tag embeddings, and hand-crafted features of all the tokens in the sliding window

[11] Using a separate classifier per contract element type makes it easier to support new contract element types. We use binary classes (positive, negative) instead of Begin, Inside, Outside (for tokens at the beginning, inside, or outside of contract elements, respectively), because consecutive contract elements are rare. We employ the SCIKIT-LEARN implementations of LR and SVM (http://scikit-learn.org/).

(e.g., 11 tokens × (200 + 25 + 17) = 2,662 features). In this case, the hand-crafted features do not include the 45 binary features that indicate the POS tag of each token, since the POS tag embeddings are also included (and hopefully provide more information than the 45 binary features). We also experimented with a version of SW-LR-ALL that included the 45 binary POS tag features per token, but the results were the same or worse, and we do not report the results of that version to save space. A sixth method, **SW-SVM-ALL**, is the same as SW-LR-ALL, except that it uses linear SVM classifiers.

4.2 Manually Written Post-Processing Rules

Some frequent errors of the sliding-window classifiers of Section 4.1 can be easily fixed by applying simple manually written post-processing rules. For example, a classifier that aims to detect effective dates may have classified correctly (in its positive class) the three tokens 'October 23 ,' (the comma is a separate token) of Fig. 2, but may have misclassified the subsequent token '2013'. A manually written post-processing written rule could check (using regular expressions) if a year-like token (e.g., '2013') that has not been classified as effective date (that has been classified as negative) follows tokens classified as effective dates (positive) that do not include a year, and change the decision of the effective dates classifier (to positive) for the year-like token. The manually written post-processing rules that we use in our experiments also group sequences of positive tokens (Fig. 2) and accept or reject groups of tokens, using 'validation' regular expressions designed to reject frequent false positive groups. The post-processing rules were constructed by inspecting and experimenting with contracts that are not included in the test part of the labeled dataset. When using manually written post-processing rules, we append the **suffix '-POST'** to the name of the method (e.g., SW-LR-EMB-POST).

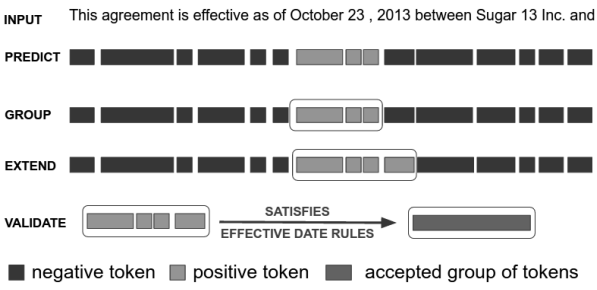

Figure 2: Example of applying post-processing rules to the decisions of a classifier.

Unfortunately, we cannot release the post-processing rules, because they also examine the characters of the (non-encoded) tokens.

4.3 Rule-Based Extraction and NER Baseline

We also compare against a previous in-house contract element extractor, named **RULES**, that relies entirely on manually crafted rules, i.e., does not use any machine learning. Its rules were also developed by inspecting and experimenting with contracts that are not included in the test part of the labeled dataset.

In the case of contracting parties, we also use **spaCy**'s NER (Section 2.2, without retraining) as a baseline in some experiments. We treat all the phrases that spaCy annotates as persons and organizations as predicted contracting parties.

5 EXPERIMENTAL RESULTS

We performed two groups of experiments, discussed in turn below.

5.1 Evaluation per Token

In a first group of experiments, we evaluated the sliding window contract element extraction methods (Section 4.1) by considering their decisions *per token*. For each contract element type (e.g., contracting parties), we measured the performance of each method in terms of *precision* ($P = \frac{TP}{TP+FP}$), *recall* ($R = \frac{TP}{TP+FN}$), and F_1 score ($F_1 = \frac{2 \cdot P \cdot R}{P+R}$). In this case, true positives (*TP*) are the tokens correctly classified as parts of contract elements of the considered type (e.g., correctly classified as positive tokens of contracting parties), *false positives* (*FP*) are the tokens incorrectly classified as parts of contract elements of the considered type (incorrectly classified as positive tokens), and *false negatives* (*FN*) are the tokens incorrectly classified as not parts of contract elements of the considered type (incorrectly classified as negative tokens). F_1 is the harmonic mean of precision and recall. All three measures are widely used in classification.

Table 4 lists the results of this group of experiments. The best results per row are shown in bold. The *macro-averages* are the averages of the corresponding columns. The macro-averaged F_1 scores show that the best methods overall are the ones that use both the hand-crafted features (Section 3.2) and the word and POS tag embeddings (Section 3.3), i.e., methods SW-LR-ALL (macro-averaged $F_1 = 0.80$) and SW-SVM-ALL (0.80). The F_1 scores of these two methods are close for most individual contract element types, with the largest differences observed in effective dates (F_1 0.67 vs. 0.72) and termination dates (F_1 0.76 vs. 0.69). SW-LR-ALL has higher macro-averaged precision (0.79 vs. 0.76), whereas SW-SVM-ALL obtained higher macro-averaged recall (0.86 vs. 0.82). Contract value and period are the most difficult contract element types (best F_1 0.64 and 0.67, respectively), possibly because the expressions that specify them are less standardized. They are also among the contract element types with the fewest tokens in the labeled dataset (Table 2), hence also with the fewest training tokens.

The methods that use only hand-crafted features (SW-LR-HCF, SW-SVM-HCF) obtain top precision results in several contract element types, especially SW-LR-HCF, but never excel in terms of recall, which indicates that the main weakness of hand-crafted features is their coverage. Although the methods that use only embeddings (SW-LR-EMB, SW-SVM-EMB) obtain very few top scores across the contract element types, we note that their F_1 scores are very close to (and often higher than) the F_1 scores of the methods that use only hand-crafted features (SW-LR-HCF, SW-SVM-HCF), which is particularly interesting given that the former two methods require almost no feature engineering (e.g., deciding which features to use).

For all the sliding window classifiers (Section 4.1), the size of the sliding window was tuned by training on 80% of the training dataset and using 20% of the training dataset as a validation set. The selected window size was almost always 13, with the exception of contracting parties and governing law where it was 11. The regularization hyper-parameters of the learning algorithms (LR, linear SVM) were tuned by performing a 3-fold cross-validation

Contract Element Type	SW-LR-EMB			SW-SVM-EMB			SW-LR-HCF			SW-SVM-HCF			SW-LR-ALL			SW-SVM-ALL		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Title	0.89	0.86	0.88	0.88	0.86	0.87	0.86	0.88	0.87	0.92	0.86	0.89	**0.91**	**0.91**	**0.91**	**0.91**	**0.91**	**0.91**
Parties	0.89	0.81	0.85	0.89	0.84	0.87	**0.94**	0.80	0.86	**0.94**	0.83	0.88	0.92	0.85	**0.89**	0.92	**0.87**	**0.89**
Start	0.77	0.95	0.85	0.76	**0.96**	0.85	**0.79**	0.94	0.86	0.77	0.92	0.83	**0.79**	**0.96**	**0.87**	0.78	**0.96**	0.86
Effective	0.63	0.61	0.62	0.60	**0.79**	0.68	**0.89**	0.45	0.60	0.86	0.74	**0.80**	0.71	0.63	0.67	0.67	**0.79**	0.72
Termination	**0.68**	0.86	0.76	0.55	0.94	0.69	0.67	0.61	0.64	0.59	0.74	0.66	**0.68**	0.86	**0.76**	0.54	**0.95**	0.69
Period	0.60	0.70	0.65	0.56	0.82	**0.67**	**0.63**	0.52	0.57	0.52	0.66	0.58	0.61	0.74	**0.67**	0.55	**0.83**	0.66
Value	0.70	0.56	0.62	0.69	0.60	**0.64**	**0.71**	0.41	0.52	0.59	0.46	0.52	0.70	0.56	0.62	0.68	**0.61**	**0.64**
Gov. Law	0.91	0.90	0.91	0.91	0.96	0.93	0.91	0.90	0.91	**0.92**	0.93	0.92	**0.92**	0.96	**0.94**	0.91	**0.97**	**0.94**
Jurisdiction	0.87	0.77	0.82	0.83	0.81	0.82	0.92	0.59	0.72	**0.93**	0.64	0.75	0.86	0.77	0.81	0.82	**0.82**	**0.82**
Legisl. Refs.	0.83	0.78	0.80	0.81	0.85	0.83	**0.87**	0.78	0.82	0.85	0.85	0.85	0.84	0.83	0.83	0.83	**0.88**	**0.86**
Headings	0.62	0.86	0.72	0.68	0.91	0.78	0.68	0.91	0.78	0.68	0.91	0.78	**0.71**	**0.92**	**0.80**	**0.71**	**0.92**	**0.80**
Macro-average	0.76	0.79	0.77	0.74	0.85	0.78	**0.81**	0.71	0.74	0.78	0.78	0.77	0.79	0.82	**0.80**	0.76	**0.86**	**0.80**

Table 4: Precision (P), Recall (R), and F1 score, *measured per token.*

on the 80% of the training dataset. Once the window size and regularization hyper-parameter values had been selected, all the classifiers were trained on the entire training dataset.

5.2 Evaluation per Contract Element Instance

Having established that the best results of the sliding window classifiers are obtained using both the hand-crafted features and the embeddings, we performed additional experiments to evaluate the effect of the post-processing rules (Section 4.2) on the best sliding window methods (SW-LR-ALL, SW-SVM-ALL) and to compare against the rule-based extractor (RULES) and SPACY's NER (Section 4.3). In these experiments, the methods were evaluated by considering their decisions *per contract element instance*. In the case of sliding window classifiers, each (maximal) sequence of consecutive predicted positive tokens (e.g., consecutive tokens predicted to be parts of contracting parties) is treated as a single predicted contract element instance, and similarly for the gold annotations of the tokens (e.g., 'Sugar 13 Inc.' in Fig. 1 is a single gold contracting party instance). Recall that the post-processing rules also produce contract element instances (groups of tokens) of this kind; RULES and spaCy's NER are also designed to identify instances of the same kind.

For each contract element type (e.g., contracting parties), the strictest evaluation would now count as true positives only the predicted contract element instances (of the particular type) that match *exactly* gold contract element instances, and similarly for false positives and false negatives. For example, if a method produced the instance 'Sugar 13' in Fig. 1, missing the 'Inc.' token, the predicted 'Sugar 13' instance would be a false positive and the gold 'Sugar 13 Inc.' instance would be a false negative. In many practical applications, however, it suffices if an element extraction method produces instances that are *almost* the same as the gold ones, especially in long instances (e.g., it does not matter if a method misses 1-2 tokens of a long title or legislation reference). Hence, we set a threshold $t \in [0.8, 1.0]$ for each contract element type (based on requirements of our clients), and we consider a predicted instance as true positive (*TP*) if (1) it is a substring of a gold instance (of the same type) and the length of the predicted instance (in characters, excluding white spaces) is at least t% of the length of the gold instance, or (2) a gold instance is a substring of the predicted instance and the length of the gold instance is at least t% of the length of the predicted instance; otherwise the predicted instance is a false positive (*FP*), and the gold instance is a false negative (*FN*) unless the gold instance matches another predicted instance.[12] Precision, recall, and F_1 are then defined as in Section 5.1, but using the new definitions of *TP*, *FP*, *FN* for contract element instances, not tokens. Unfortunately, to determine if a predicted and a gold contract element instance satisfy the character length thresholds, one has to examine the characters of the (non-encoded) tokens of the test contracts, which is impossible with the encoded datasets we provide. Hence, only token-based evaluation (Table 4) is possible with the encoded datasets.

Table 5 lists the results of our second group of experiments, where the methods were evaluated per contract element instance. The results of Table 5 are not directly comparable to those of Table 4, where the evaluation was per token. Nevertheless, Table 5 shows that the post-processing rules (-POST) clearly improve the performance of SW-LR-ALL and SW-SVM-ALL (in both cases, from 0.69 macro-averaged F_1 to 0.86). The most dramatic improvements are observed in legislation references (from 0.36 and 0.27 macro-averaged F_1 to 0.92 and 0.94) and effective dates (from 0.48 to 0.83 and 0.91), but significant improvements are also observed in several other element types (e.g., termination dates, contract values, jurisdiction, clause headings), indicating that the post-processing rules correct many frequent errors of the sliding window classifiers. The extractor that uses only manually written rules (RULES) performs overall better (0.74 vs. 0.69 macro-averaged F_1) than the methods that rely only on machine learning (SW-LR-ALL, SW-SVM-ALL), achieving top precision or recall scores for several contract element types. The methods that combine machine learning and post-processing rules (SW-LR-ALL-POST, SW-SVM-ALL-POST), however, are overall better, which suggests that manually writing post-processing rules to correct the decisions of machine learning classifiers may be a better investment of effort than developing entirely rule-based extractors.

In the case of contracting parties, the spaCy NER baseline (results in brackets in Table 5) obtains lower precision, higher recall, and almost the same F_1 score compared to RULES, but performs clearly worse than the sliding window classifiers (with or without post-processing). The latter indicates that training classifiers

[12] The values of t are: 1.0 for start dates, effective dates, termination dates; 0.9 for governing law and clause headings; 0.8 for other contract element types.

Contract Element Type	RULES			SW-LR-ALL			SW-SVM-ALL			SW-LR-ALL-POST			SW-SVM-ALL-POST		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Title	**0.97**	0.54	0.69	0.86	0.86	0.86	0.88	0.86	0.87	0.84	**0.93**	**0.88**	0.83	**0.93**	**0.88**
Parties (spaCy)	**0.96** (0.53)	0.39 (0.62)	0.56 (0.57)	0.93	0.80	0.86	0.90	0.82	0.86	**0.96**	**0.85**	**0.90**	0.95	**0.85**	0.89
Start	0.68	0.88	0.77	0.68	0.89	0.77	0.84	0.93	0.88	**0.89**	**0.94**	**0.91**	0.84	0.93	0.88
Effective	0.84	**0.95**	0.89	0.57	0.41	0.48	0.44	0.53	0.48	0.86	0.80	0.83	**0.87**	**0.95**	**0.91**
Termination	0.66	**0.98**	0.79	0.52	0.74	0.61	0.49	0.85	0.62	**0.79**	0.91	**0.85**	0.75	0.96	0.84
Period	0.14	0.82	0.24	0.53	0.73	0.61	0.45	0.73	0.56	**0.62**	**0.85**	**0.72**	0.51	0.80	0.63
Value	0.69	0.92	0.79	0.65	0.86	0.74	0.67	0.86	0.75	**0.74**	0.92	**0.82**	0.72	**0.94**	0.81
Gov. Law	0.97	0.90	0.93	0.96	0.90	0.93	0.94	0.91	0.92	**0.99**	0.93	0.96	**0.99**	**0.95**	**0.97**
Jurisdiction	**0.99**	0.59	0.74	0.84	0.72	0.78	0.65	**0.84**	0.74	**0.99**	0.75	0.85	0.98	0.78	**0.87**
Legisl. Refs.	**0.97**	**0.91**	**0.94**	0.23	0.83	0.36	0.16	0.87	0.27	**0.97**	0.88	0.92	**0.97**	0.90	**0.94**
Headings	0.80	**0.86**	0.83	0.58	0.70	0.64	0.60	0.71	0.65	**0.94**	0.80	**0.86**	**0.94**	0.80	**0.86**
Macro-average	0.79 (0.75)	0.80 (0.82)	0.74 (0.74)	0.67	0.77	0.69	0.64	0.81	0.69	**0.87**	0.87	**0.86**	0.85	**0.89**	**0.86**

Table 5: Precision (P), Recall (R), and F1-score, *measured per contract element instance.*

especially for contract element extraction is better than using a generic NER without retraining (and without modifying its features), even for contracting parties that are close to named entity types (organizations, persons) typically supported by generic NERs.

6 RELATED WORK

Curtotti et al. [5] classified lines (separated by line breaks) of Australian contracts into 32 classes. Two of their classes correspond to our contract titles and clause headings. Two of their other classes correspond to entire lines containing contract elements we aim to detect: 'partyline' is a line that contains a contracting party, without tagging the exact tokens of the party; 'datemadeline' is presumably a line containing a start date. There is no correspondence between the other 28 classes of Curtotti and Mccreath (e.g., 'recitalhead', 'recitalline', 'contactofficer', 'emailline') and the other contract elements of our work (e.g., termination date, governing law). Curtotti and Mccreath experimented with several machine learning algorithms (e.g., SVM, decision trees), using 40 hand-crafted features. They obtained their best results (83.48% accuracy) with a single multi-class classifier that combined machine learning (Random Forest) and manually written tagging rules (which provided additional features to the Random Forest). They experimented, however, with only 30 contracts from a corpus of 256.

Indikuri et al. [12] employed SVMs and n-gram features to classify contract sentences as clauses or non-clauses, and classify clauses as payment terms or not, experimenting with only 73 sentences. Gao et al. [8] used 2,647 contracts, but experimented only with patterns to detect exception clauses (e.g., "in case of defect").[13] We are unaware of other legal text analytics work on contracts.

In the broader context of legal text analytics Stranieri et al. [28], Francesconi et al. [7], Mencia et al. [20] used an SVM and hand-crafted features to segment French laws (e.g., identify titles, articles), experimenting with 181 texts (1,146 articles). Hasan et al. [11] relied on heuristics to segment Spanish legislative bulletins into their components (e.g., articles), assuming that each bulletin includes a table of contents, and experimenting with 50 texts. Biagioli et al. [2] used an SVM with bag-of-word features to detect paragraphs of Italian laws with particular types of information (e.g., obligation, sanction), then pattern matching to fill in type-specific slots (e.g., entity sanctioned), experimenting with 582 paragraphs.

Dozier et al. [6] identified judges, attorneys, companies, jurisdictions, and courts in US trial documents. A CRF [13] with n-gram, positional, and punctuation features was used to segment each document into zones; then lists of known entities (e.g., courts) and hand-crafted patterns were used to extract named entities from particular zones. Manually constructed rules were employed to map each extracted entity to a record (e.g., containing fields for the first name and surname of an extracted attorney name, along with city names that occurred near the attorney name) and retrieve candidate matching records from authority files (e.g., records of known attorneys). An SVM with field-specific similarity measures as features was subsequently used to select the 'best' authority file record per extracted named entity record.

Quaresma et al. [27] employed an SVM with TF-IDF bag-of-word features to classify European international agreements per topic. They also used manually crafted patterns operating on parse trees to extract locations, organizations, dates, and document references. The experiments were performed on 2,714 agreements, each in four languages (English, German, Italian, Portuguese).

To summarize, previous legal text analytics work on contracts has focused on classifying entire lines, sentences, or clauses, using smaller datasets or fewer classes. In the broader legal text analytics context, the closest related work has considered segmenting legal (mostly legislative) documents and recognizing named entities, but the proposed methods are not directly applicable to contract element extraction. For example, they employ hand-crafted features, patterns, or lists of known entities that would have to be tailored for contracts. Also, none of the previous work discussed above considered word (and POS tag) embeddings.

7 CONCLUSIONS AND FUTURE WORK

We considered contract element extraction, a legal text analytics task with significant practical value. As a starting point, we constructed and made publicly available a labeled dataset of approximately 3,500 English contracts with gold contract element annotations, which can be used to train and test contract element extraction methods, along with a larger unlabeled dataset of approximately 750,000 English contracts, which can be used to pre-train

[13] The contracts of Gao et al. were obtained from http://contracts.onecle.com/.

word and POS tag embeddings. To bypass privacy issues, both datasets are provided in an encoded form, where each vocabulary word has been replaced by an integer identifier. Word and POS tag embeddings (pre-trained on the unlabeled dataset) and hand-crafted features are also provided for the tokens of the datasets.

We experimented with contract element extraction methods that rely on linear classifiers (LR, linear SVM) with hand-crafted features, word and POS tag embeddings. We also considered manually written rules used instead of the linear classifiers or to post-process their decisions. A first group of experiments showed that the linear classifiers performed best when both the hand-crafted features and the word and POS tag embeddings were used. Interestingly, the embeddings on their own (without any hand-crafted features) led to very similar, though overall inferior performance. In a second group of experiments, we studied the effect of manually written post-processing rules that correct frequent errors of the linear classifiers; we also compared against an entirely rule-based system and a generic NER (for contracting parties only). The post-processing rules significantly improved the performance of the linear classifiers, leading to the same overall results for both LR and SVM, outpeforming the rule-based system and the generic NER. The F_1 score of the two best systems exceeded 0.84 (measured per contract element instance) in all but one contract element types. We view the methods of this paper as strong baselines for further work that may experiment with more complex classifiers (e.g., convolutional or recurrent neural networks [9, 10]) using the data we provide.

REFERENCES

[1] Ion Androutsopoulos, John Koutsias, Konstantinos V. Chandrinos, and Constantine D. Spyropoulos. 2000. An Experimental Comparison of Naive Bayesian and Keyword-Based Anti-Spam Filtering with Encrypted Personal E-mail Messages. In *Proceedings of the 23rd ACM SIGIR Conference*. Athens, Greece, 160–167.

[2] Carlo Biagioli, Enrico Francesconi, Andrea Passerini, Simonetta Montemagni, and Claudia Soria. 2005. Automatic Semantics Extraction in Law Documents. In *Proceedings of the 10th International Conference on Artificial Intelligence and Law*. Bologna, Italy, 133–140.

[3] Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. 1999. An Algorithm That Learns What's in a Name. *Machine Learning* 34, 1–3 (1999), 211–231.

[4] Nello Cristianini and John Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press.

[5] Michael Curtotti and Eric Mccreath. 2010. Corpus Based Classification of Text in Australian Contracts. In *Proceedings of the Australasian Language Technology Association Workshop*. Melbourne, Australia, 18–26.

[6] Christopher Dozier, Ravikumar Kondadadi, Marc Light, Arun Vachher, Sriharsha Veeramachaneni, and Ramdev Wudali. 2010. Named Entity Recognition and Resolution in Legal Text. In *Semantic Processing of Legal Texts*, Enrico Francesconi, Simonetta Montemagni, Wim Peters, and Daniela Tiscornia (Eds.). Number 6036 in Lecture Notes in AI. Springer.

[7] Enrico Francesconi, Simonetta Montemagni, Wim Peters, and Daniela Tiscornia. 2010. *Semantic Processing of Legal Texts*. Number 6036 in Lecture Notes in AI. Springer.

[8] Xibin Gao, Munindar P. Singh, and Pankaj Mehra. 2012. Mining Business Contracts for Service Exceptions. *IEEE Transactions on Services Computing* 5 (2012), 333–344.

[9] Yoav Goldberg. 2016. A Primer on Neural Network Models for Natural Language Processing. *Journal of Artificial Intelligence Research* 57 (2016), 345–420.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press.

[11] Ismael Hasan, Javier Parapar, and Roi Blanco. 2008. Segmentation of Legislative Documents Using a Domain-Specific Lexicon. In *Proceedings of the 19th International Conference on Database and Expert Systems Application*. Turin, Italy, 665–669.

[12] Kishore Varma Indukuri and P. Radha Krishna. 2010. Mining e-Contract Documents to Classify Clauses. In *Proceedings of the 3rd Annual ACM Bangalore Conference*. Bangalore, India, Article 7, 5 pages.

[13] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 18th International Conference on Machine Learning*. Williamstown, MA, 282–289.

[14] Omer Levy and Yoav Goldberg. 2014. Linguistic Regularities in Sparse and Explicit Word Representations. In *Proceedings of the 18th Conference on Computational Natural Language Learning*. Baltimore, MD, 171–180.

[15] Omer Levy and Yoav Goldberg. 2014. Neural Word Embedding As Implicit Matrix Factorization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*. Montreal, Canada, 2177–2185.

[16] Omer Levy, Yoav Goldberg, and Ido Dagan. 2016. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics* 3 (2016), 211–225.

[17] Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, 1520–1530.

[18] Thang Luong, Richard Socher, and Christopher Manning. 2013. Better Word Representations with Recursive Neural Networks for Morphology. In *Proceedings of the 17th Conference on Computational Natural Language Learning*. Sofia, Bulgaria, 104–113.

[19] Peter McCullagh and John A. Nelder. 1989. *Generalized Linear Models, (Chapman & Hall/CRC Monographs on Statistics & Applied Probability)* (2 ed.). Chapman and Hall/CRC.

[20] Eneldo Loza Mencia. 2009. Segmentation of legal documents. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law*. Barcelona, Spain, 88–97.

[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of the International Conference on Learning Representations (ICLR)*. Scottsdale, AZ.

[22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*. Stateline, NV.

[23] Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, GA, 746–751.

[24] Z. Milosevic, S. Gibson, P. F. Linington, J. Cole, and S. Kulkarni. 2004. On Design and Implementation of a Contract Monitoring Facility. In *Proceedings of the 1st IEEE International Workshop on Electronic Contracting*. IEEE Computer Society Press, San Diego, CA, 62–70.

[25] David Nadeau and Satoshi Sekine. 2007. A Survey of Named Entity Recognition and Classification. *Lingvisticae Investigationes* 30, 1 (2007), 3–26.

[26] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar, 1532–1543.

[27] Paulo Quaresma and Teresa Gonçalves. 2010. Using Linguistic Information and Machine Learning Techniques to Identify Entities from Juridical Documents. In *Semantic Processing of Legal Texts*, Enrico Francesconi, Simonetta Montemagni, Wim Peters, and Daniela Tiscornia (Eds.). Number 6036 in Lecture Notes in AI. Springer, 44–59.

[28] Andrew Stranieri and John Zeleznikow. 2005. *Knowledge Discovery from Legal Databases*. Springer.

[29] Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag.

[30] Hsiang-Fu Yu, Fang-Lan Huang, and Chih-Jen Lin. 2011. Dual Coordinate Descent Methods for Logistic Regression and Maximum Entropy Models. *Machine Learning* 85, 1-2 (2011), 41–75.