Legal Knowledge and Information Systems
A. Wyner and G. Casini (Eds.)

doi:10.3233/978-1-61499-838-9-155

A Deep Learning Approach to Contract Element Extraction

Ilias Chalkidis [a,b], Ion Androutsopoulos [a]
[a] *Athens University of Economics and Business, Greece*
[b] *Cognitiv+ Ltd., London, UK*

Abstract. We explore how deep learning methods can be used for contract element extraction. We show that a BILSTM operating on word, POS tag, and token-shape embeddings outperforms the linear sliding-window classifiers of our previous work, without any manually written rules. Further improvements are observed by stacking an additional LSTM on top of the BILSTM, or by adding a CRF layer on top of the BILSTM. The stacked BILSTM-LSTM misclassifies fewer tokens, but the BILSTM-CRF combination performs better when methods are evaluated for their ability to extract entire, possibly multi-token contract elements.

Keywords. Natural language processing, deep learning, legal text analytics.

1. Introduction

Law firms, companies, government agencies etc. need to monitor contracts for a wide range of tasks [1]. For example, law firms need to inform their clients when contracts are about to expire or when they are affected by legislation changes. Contractors need to keep track of agreed payments and deliverables. Law enforcement agencies 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.

We recently released a benchmark dataset of approximately 3,500 English contracts, annotated with 11 types of contract elements, the largest publicly available dataset for contract element extraction.[1] Using that dataset, in previous work [2] we experimented with Logistic Regression [3] and linear Support Vector Machines (SVMs) [4], both operating on fixed-size sliding windows of tokens, represented using hand-crafted features, pre-trained word embeddings [5,6], and/or pre-trained part-of-speech (POS) tag embeddings. We also experimented with manually written rules that replaced the machine learning classifiers or post-processed their decisions. In this paper, we experiment with deep learning methods [7,8] on the same dataset. We show that a bidirectional LSTM (BILSTM) [9,10,11] operating on word, POS tag, and token-shape embeddings outperforms the best methods of our previous work, in most cases, without using any manually written rules. Further improvements are observed by stacking an additional LSTM on top of the BIL-

[1] The dataset is available from `http://nlp.cs.aueb.gr/publications.html`.

STM [12,13] or by adding a Conditional Random Field (CRF) layer [14] on top of the BILSTM [15,16,17]. The stacked BILSTM-LSTM misclassifies fewer tokens, but the BILSTM-CRF combination performs better when methods are evaluated for their ability to extract entire, possibly multi-token contract elements.

2. Contract Element Extraction Methods

The dataset of our previous work specifies particular *extraction zones* for each contract element type [2]. For example, contracting parties are to be extracted from the cover page and preamble of each contract, whereas legislation references are to be extracted from zones starting up to 20 tokens before and ending up to 20 tokens after each occurrence of words like "act" or "treaty" in the main text. The extraction zones are explicitly marked in each training and test contract of the dataset, and can be easily produced in practice (e.g., using regular expressions). We use the same extraction zones in this paper.

We also use the pre-trained 200-dimensional word embeddings and 25-dimensional POS tag embeddings that accompany the dataset [2], which were obtained by applying WORD2VEC (skip-gram model) [18] to approximately 750,000 unlabeled and 50,000 POS-tagged English contracts, respectively. We also use 5-dimensional *token shape embeddings* that represent the following seven possible shapes of tokens: token consisting of alphabetic upper case characters, possibly including periods and hyphens (e.g., 'AGREEMENT', 'U.S.', 'CO-OPERATION'); token consisting of alphabetic lower case characters, possibly including periods and hyphens (e.g., 'registered', 'etc.', 'third-party'); token with at least two characters, consisting of an alphabetic upper case first character, followed by alphabetic lower-case characters, possibly including periods and hyphens (e.g., 'Limited', 'Inc.', 'E-commerce'); token consisting of digits, possibly including periods and commas (e.g., '2009', '12,000', '1.1'); line break; any other token containing only non-alphanumeric characters (e.g., '$', '##'); any other token (e.g., '3rd', 'strangeTek', 'EC2'). The token shape embeddings were obtained by applying WORD2VEC (again, skip-gram model, same other settings) to approx. 2,000 contracts of the unlabeled dataset of our previous work [2], after replacing the tokens by pseudo-tokens (e.g., 'allupper', 'alllower') reflecting the corresponding token shape.[2]

As already noted, in our previous work we experimented with linear classifiers (Logistic Regression and linear SVMs) operating on fixed-size sliding windows of tokens. We also experimented with manually written rules that replaced the linear classifiers or post-processed their decisions. Those methods are described in detail in our previous work [2]. Below we describe the new, LSTM-based methods we experimented with. We do not consider manually written rules in any detail, because in most cases the LSTM-based methods outperform our previous methods (with or without rules) without employing any rules. As in our previous work, for each LSTM-based method we build a separate extractor for each contract element type (e.g., contracting parties), 11 extractors in total per method, which allows us to compare directly against our previous results.[3]

[2]To by-pass privacy issues, in the dataset each token is replaced by a unique integer identifier, but hand-crafted features, word, and POS tag embeddings are still provided for each token [2]. The new token shape embeddings will also be made publicly available. We use no hand-crafted features in this paper.

[3]The LSTM-based methods were implemented using KERAS (`https://keras.io/`) with a TENSORFLOW backend (`https://www.tensorflow.org/`).

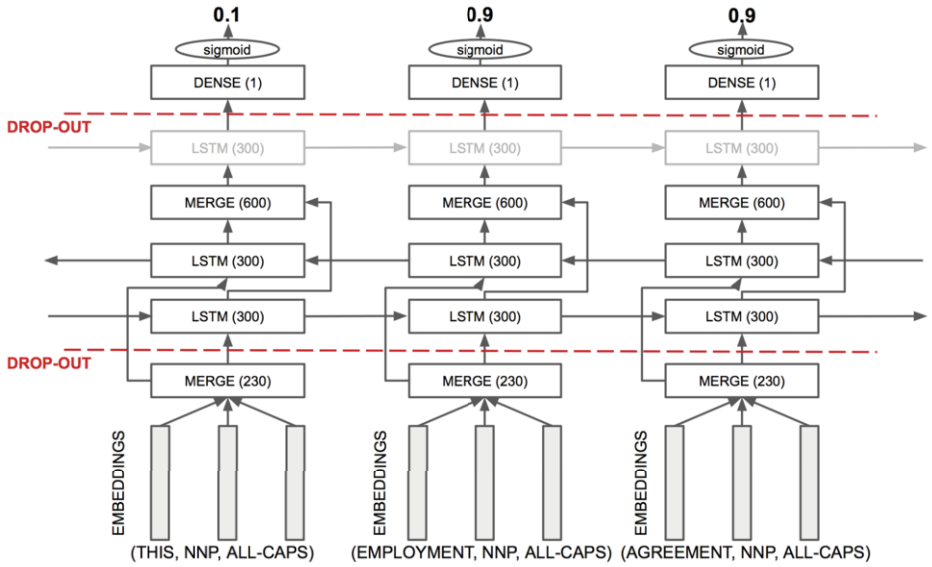

Figure 1. BILSTM-(LSTM)-LR extractor for a particular contract element type.

2.1. BILSTM-LR Extractors

In the first LSTM-based method, called BILSTM-LR, each extractor (Fig. 1, without the upper LSTM boxes) uses its own bidirectional LSTM (BILSTM) chain [12] to convert the concatenated word, POS tag, and token shape embeddings of each token (lower MERGE boxes) of an extraction zone to context-aware token embeddings (upper MERGE boxes). Each context-aware token embedding is then passed on to a Logistic Regression (LR) layer (DENSE boxes and sigmoid ovals) to estimate the probability that the corresponding token is positive (e.g., part of a contracting party element) with respect to the contract element type of the particular extractor.

We use 300-dimensional hidden states in both LSTM chains. Larger dimensionalities slow down our experiments, without noticeable efficacy improvements. We employ LSTM cells with input, forget, and output gates [9,10,19], with DROPOUT [20] after the merged embeddings and before the LR layer (Fig. 1). We used Glorot initialization [21], binary cross-entropy loss, and the Adam optimizer [22] to train each BILSTM-LR extractor, with early stopping examining the validation loss. The DROPOUT rate, learning rate, and batch size (possibly different per contract element type) were tuned performing a 3-fold cross-validation on 80% of the training extraction zones (of the corresponding contract element type), using one third of the 80% of the training extraction zones as a validation set in each fold. Having selected DROPOUT rate, learning rate, and batch size (per contract element type), each BILSTM-LR extractor was re-trained on the entire 80% of the training extraction zones, using the remaining 20% as a validation set. Out of vocabulary words, meaning words we had no pre-trained embeddings for, were mapped to random embeddings, as in our previous work. At test time, each token is classified as positive if the corresponding probability of the LR layer (Fig. 1) exceeds 0.5.

2.2. BILSTM-LSTM-LR Extractors

The second LSTM-based method, BILSTM-LSTM-LR, is the same as the previous one, except that it has an additional LSTM chain (upper LSTM boxes in Fig. 1) between the context-aware token embeddings (MERGE (600) boxes) of the lower BILSTM chain, and the logistic regression (LR) layer (DENSE boxes and sigmoid ovals). Stacking LSTM (or BILSTM) chains has been reported to improve efficacy in several linguistic tasks [13,23] at the expense of increased computational cost. To reduce the computational cost, it is common to make the stacked LSTM chains unidirectional, rather than bidirectional [23]. Hyper-parameter tuning and training are performed as in BILSTM-LR (Section 2.1).

2.3. BILSTM-CRF Extractors

In the third LSTM-based method, BILSTM-CRF, we replace the upper LSTM chain and the LR layer of the BILSTM-LSTM-LR extractor (upper LSTM and DENSE boxes, sigmoid ovals of Fig. 1) by a linear-chain Conditional Random Field (CRF). CRFs [14] have been widely used in sequence labeling (e.g., POS tagging, named entity recognition). They have also shown promising results on top of LSTM, BILSTM, or feed-forward neural network layers in sequence labeling [24,25,15,16,17] and parsing [26]. In our case, the CRF layer jointly selects the assignment of positive or negative labels to the entire token sequence of an extraction zone, which allows taking into account the predicted labels of neighboring tokens. For example, if both the previous and the next token of the current token are classified as parts of a legislation reference, this may be an indication that the current token is also part of the same legislation reference.

Again, we train a separate BILSTM-CRF extractor per contract element type. Training combines dynamic programming or beam search decoding with backpropagation to maximize log-likelihood [25,26].[4] Hyper-parameter tuning and training are performed as in the previous methods. We note that in tasks with richer sets of labels, as opposed to our only two labels ('positive', 'negative'), a CRF layer may be more beneficial. For example, in POS tagging [15,17] a CRF layer can learn that a determiner is usually followed by an adjective or noun, rather than a verb. One way to enrich our label set would be to use a single classifier for all the contract element types. This would be complicated, however, by the fact that contract elements of different types may have different extraction zones.

3. Experimental Results

We performed two groups of experiments, reported in turn below, where the methods of Section 2 where evaluated per token and contract element, respectively.

3.1. Evaluation per Token

In the first group of experiments, we evaluated the methods by considering their decisions *per token*. For each contract element type (e.g., contracting parties), we measured the

[4] We use the CRF layer implementation of KERAS-CONTRIB (`https://github.com/farizrahman4u/keras-contrib/blob/master/keras_contrib/layers/crf.py`), with joint conditional log-likelihood optimization and Viterbi best path prediction (decoding).

ELEMENT	SW-LR-ALL			SW-SVM-ALL			BILSTM-LR			BILSTM-LSTM-LR			BILSTM-CRF		
TYPE	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Title	0.91	0.91	0.91	0.91	0.91	0.91	0.95	0.93	0.94	0.94	**0.95**	**0.95**	**0.96**	**0.95**	**0.95**
Parties	0.92	0.85	0.89	0.92	0.87	0.89	0.97	0.92	**0.95**	0.97	**0.94**	**0.95**	**0.98**	0.92	**0.95**
Start	0.79	0.96	0.87	0.78	0.96	0.86	0.91	0.97	0.94	**0.93**	0.97	**0.95**	0.92	**0.98**	**0.95**
Effective	0.71	0.63	0.67	0.67	0.79	0.72	**0.98**	0.92	0.95	0.97	**0.96**	**0.97**	0.95	0.89	0.92
Termination	0.68	0.86	0.76	0.54	**0.95**	0.69	0.65	0.90	0.75	**0.70**	0.92	**0.79**	0.65	0.93	0.77
Period	**0.61**	0.74	**0.67**	0.55	0.83	0.66	0.44	0.82	0.57	0.47	**0.86**	0.59	0.55	0.85	0.65
Value	0.70	0.56	0.62	0.68	0.61	0.64	**0.74**	0.55	0.63	**0.74**	**0.63**	**0.68**	0.72	0.60	0.66
Gov. Law	0.92	0.96	0.94	0.91	0.97	0.94	0.98	**0.98**	0.98	**0.99**	**0.98**	**0.98**	**0.99**	0.97	**0.98**
Jurisdiction	0.86	0.77	0.81	0.82	0.82	0.82	**0.90**	**0.89**	**0.89**	**0.90**	0.88	**0.89**	**0.90**	0.88	0.88
Legisl. Refs.	0.84	0.83	0.83	0.83	0.88	0.86	0.82	**0.95**	**0.88**	0.83	0.94	**0.88**	0.82	0.94	0.87
Headings	0.71	0.92	0.80	0.71	0.92	0.80	0.98	0.97	**0.98**	**0.99**	**0.98**	**0.98**	**0.99**	0.97	**0.98**
Macro-average	0.79	0.82	0.80	0.76	0.86	0.80	0.85	0.89	0.86	**0.86**	**0.91**	**0.87**	**0.86**	0.90	**0.87**

Table 1. Precision (P), Recall (R), and F_1 score, *measured per token*. Best results per contract element type shown in bold font in gray cells.

performance of each method in terms of *precision* ($P = \frac{TP}{TP+FP}$), *recall* ($R = \frac{TP}{TP+FN}$), and F_1 score ($F_1 = \frac{2 \cdot P \cdot R}{P+R}$). Here, true positives (*TP*) are tokens correctly classified as parts of contract elements of the considered type, *false positives* (*FP*) are tokens incorrectly classified as parts of contract elements of the considered type, and *false negatives* (*FN*) are tokens incorrectly classified as not parts of contract elements of the considered type. F_1 (harmonic mean) is commonly used to combine precision and recall.

Table 1 lists the results of this group of experiments. The best results per contract element type are shown in bold font in gray cells. The *macro-averages* are the averages of the corresponding columns, indicating the overall performance of each method on all the contract element types. The best methods of our previous work in these experiments [2] are SW-LR-ALL and SW-SVM-ALL, which use hand-crafted features, word, and POS tag embeddings, with LR or SVM classifiers operating on fixed-size windows of tokens. Overall, both of these methods perform equally well (0.80 macro-averaged F_1).

The three LSTM-based methods overall perform clearly better than the linear sliding-window classifiers of our previous work. Even BILSTM-LR, the simplest of the three LSTM-based methods, exceeds the macro-averaged F_1 score of the best previous methods by 6 points (0.86 vs. 0.80). The extra LSTM layer of BILSTM-LSTM-LR improves the macro-averaged F_1 score by only 1 point (0.87). By looking at the results for individual contract element types, however, we see that BILSTM-LSTM-LR obtains top F_1 scores for all but one contract element types (the exception being contract periods), and for some element types (most notably, termination dates and contract values) it performs significantly better than BILSTM-LR (0.79 vs. 0.75, and 0.68 vs. 0.63 F_1, respectively). Although BILSTM-CRF has the same macro-averaged F_1 as BILSTM-LSTM-LR (0.87), it does not perform better than BILSTM-LSTM-LR in any contract element type, except for contract periods, where it outperforms BILSTM-LSTM-LR (0.65 vs. 0.59 F_1). The best F_1 for contract periods, however, is achieved by SW-LR-ALL (0.67); SW-SVM-ALL (0.66 F_1) also exceeds the F_1 score of BILSTM-CRF (0.65) for contract periods.

The lowest F_1 scores of all three LSTM-based methods are for contract periods, termination dates, and contract values, which are the three contract element types with the fewest training instances in the dataset [2]. The performance of the best sliding-window methods (SW-LR-ALL, SW-SVM-ALL) is close to or better than the performance of BILSTM-LR (the weakest LSTM-based method) in these three contract element types;

and both BILSTM-LSTM-LR and BILSTM-CRF show some of their biggest improvements compared to the simpler BILSTM-LR in these three types.

It seems that the LSTM-based methods perform poorly for contract element types with few training instances, to the extent that the best linear sliding-window classifiers are able to catch up. Nevertheless, the extra layer of BILSTM-LSTM-LR and the CRF layer of BILSTM-CRF are particularly beneficial in contract element types with few training instances, leading to significant performance improvements compared to BILSTM-LR. We can only speculate that the additional LSTM layer of BILSTM-LSTM-LR may lead to better generalization, and that the CRF layer may in effect introduce an additional training signal by allowing BILSTM-CRF to consider more directly the predicted labels of neighboring tokens.

3.2. Evaluation per Contract Element

In the second group of experiments, the methods were evaluated for their ability to identify *entire contract elements*. By contrast, the linear sliding-window classifiers and the LSTM-based methods classify individual tokens as positive or negative with respect to a particular contract element type. For the experiments of this section, each (maximal) sequence of consecutive tokens predicted to be positive with respect to a contract element type (e.g., consecutive tokens predicted to be parts of contracting parties) is treated as a single predicted contract element of the corresponding type (e.g., a single contracting party), and similarly for the gold annotations, as in our previous work [2].

For each contract element type (e.g., contracting parties), the strictest evaluation would now count as true positives only the predicted contract elements that match *exactly* gold ones, and similarly for false positives and false negatives. For example, if a method predicted "Sugar 13" to be a contracting party, missing the "Inc." of the gold "Sugar 13 Inc.", the predicted 'Sugar 13' would be a false positive and the gold "Sugar 13 Inc." would be a false negative. In practical applications, however, it often suffices to produce contract elements that are *almost* the same as the gold ones. As in our previous work, we set a threshold $t \in [0.8, 1.0]$ for each contract element type (based on requirements of our clients, the same as in the experiments of our previous work) and we consider a predicted contract element as true positive (*TP*) if (1) it is a substring of a gold contract element (of the same type) and the length of the predicted element (in characters, excluding white spaces) is at least $t\%$ of the length of the gold one, or vice versa (2) a gold contract element is a substring of the predicted one and the length of the gold element is at least $t\%$ of the length of the predicted one. Otherwise the predicted contract element is a false positive (*FP*); and the corresponding gold element is a false negative (*FN*), unless it matches another predicted element of the same type.[5] Precision, recall, and F_1 are then defined as in Section 3.1, but now using the definitions of *TP*, *FP*, *FN* of this paragraph.

Table 2 lists the results of the second group of experiments. We now include the manually crafted post-processing rules of our previous work [2] in the best linear sliding window classifiers, since they improve significantly their performance, as reported in our previous work.[6] The simplest LSTM-based method, BILSTM-LSTM-LR, equals the

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

[6]The post-processing rules cannot be used when evaluating per token (Section 3.1), because they require multi-token contract elements to have been grouped into single contract elements, as in this section.

Element	SW-LR-ALL-POST			SW-SVM-ALL-POST			BILSTM-LR			BILSTM-LSTM-LR			BILSTM-CRF		
Type	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Title	0.84	0.93	0.88	0.83	0.93	0.88	0.96	0.94	**0.95**	0.95	**0.95**	**0.95**	**0.97**	**0.95**	**0.95**
Parties	0.96	0.85	0.90	0.95	0.85	0.89	0.98	0.89	0.93	0.96	**0.91**	**0.94**	**0.99**	0.87	0.93
Start	0.89	0.94	0.91	0.84	0.93	0.88	0.88	**0.96**	0.92	0.92	0.95	0.93	**0.93**	**0.96**	**0.94**
Effective	0.86	0.80	0.83	0.87	0.95	0.91	**0.99**	0.89	**0.94**	0.94	0.94	**0.94**	0.95	0.86	0.90
Termination	**0.79**	0.91	**0.85**	0.75	0.96	0.84	0.73	0.89	0.80	0.72	0.91	0.81	**0.74**	**0.98**	0.84
Period	**0.62**	**0.85**	**0.72**	0.51	0.80	0.63	0.40	0.71	0.51	0.44	0.75	0.55	**0.62**	0.79	0.70
Value	0.74	0.92	**0.82**	0.72	**0.94**	0.81	0.66	0.67	0.67	**0.75**	0.72	0.73	0.70	0.78	0.74
Gov. Law	**0.99**	0.93	0.96	**0.99**	0.95	0.97	0.98	**0.97**	0.97	**0.99**	**0.97**	**0.98**	**0.99**	0.96	0.97
Jurisdiction	0.99	0.75	0.85	0.98	0.78	0.87	0.90	0.85	0.88	0.92	0.82	0.86	0.93	**0.85**	**0.89**
Legisl. Refs.	**0.97**	0.88	0.92	**0.97**	0.90	**0.94**	0.86	**0.94**	0.90	0.87	0.93	0.90	0.92	**0.94**	0.93
Headings	0.94	0.80	0.86	0.94	0.80	0.86	**0.99**	**0.89**	**0.94**	**0.99**	**0.89**	**0.94**	**0.99**	0.88	**0.94**
Macro-average	0.87	0.87	0.86	0.85	**0.89**	0.86	0.85	0.87	0.86	0.86	**0.89**	0.87	**0.88**	**0.89**	**0.88**

Table 2. Precision (P), Recall (R), and F_1 score, *measured per contract element instance*. Best results per contract element type shown in bold font in gray cells.

macro-averaged F_1 score (0.86) of the best linear sliding-window classifiers (SW-LR-ALL-POST, SW-SVM-ALL-POST), *without using any manually written rules*, unlike the sliding-window classifiers that *rely extensively* on the *post-processing rules* in these experiments; without the post-processing rules, the macro-averaged F_1 score of the best linear sliding window classifiers (SW-LR-ALL, SW-SVM-ALL) drops to 0.69 (not shown in Table 2, see our previous work). This is particularly important, because the post-processing rules are very difficult to maintain in practice, since they have to be tailored to the errors of each particular classifier per contract element type.

The extra LSTM layer of BILSTM-LSTM-LR improves the macro-averaged F_1 of BILSTM-LR by one point (0.87 vs. 0.86), but BILSTM-CRF now performs even better overall (0.88 macro-averaged F_1). By looking at the F_1 scores per contract element type, we see that BILSTM-CRF now performs better or at least as well as BILSTM-LR in all but one contract element types, the exception being effective dates where BILSTM-LR is better (0.94 vs. 0.90 F_1). In several contract element types, the improvements of BILSTM-CRF compared to BILSTM-LR are very significant, with the largest improvements observed in contract periods (from 0.51 to 0.70 F_1), contract values (from 0.67 to 0.74), and termination dates (from 0.80 to 0.84). Recall that these are the three contract element types with the fewest training instances in the dataset. As in the experiments of the previous section, they are also the contract element types where all the LSTM-based methods again obtain their lowest F_1 scores, and where the linear sliding-window classifiers catch up with or exceed (in the case of SW-LR-ALL-POST) the LSTM-based methods. The extra LSTM layer of BILSTM-LSTM-LR also improves the performance of BILSTM-LR in these three contract element types (from 0.51 to 0.55, from 0.80 to 0.81, and from 0.67 to 0.72, respectively), but the improvements are smaller compared to those of BILSTM-CRF. Like BILSTM-CRF, BILSTM-LSTM-LR improves or matches the F_1 of BILSTM-LR in all but one of the eleven contract element types, the exception being jurisdiction, where BILSTM-LR is better (0.88 vs. 0.86 F_1), but the improvements are smaller compared to those of BILSTM-CRF. Overall, BILSTM-CRF appears to be better than BILSTM-LSTM-LR in the experiments of this section, in contrast to the experiments of the previous section, suggesting that although BILSTM-LSTM-LR makes fewer errors per token, the errors of BILSTM-CRF are less severe, in the sense that the thresholds t allow more of the extracted contract elements of BILSTM-CRF to be considered as successfully extracted.

4. Related Work

Our previous work [2] provides an extensive overview of related work, concluding that previous text analytics work on contracts [27,28,29] focused on classifying entire lines, sentences, or clauses, rather than extracting specific contract elements, and used much smaller datasets or fewer classes. In the broader legal text analytics context, our previous work concludes that the closest related work considered segmenting legal (mostly legislative) documents [30,31,32,33,34,35] and recognizing named entities [36,35], 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.

More recently, Garcia-Constantino et al. [37] experimented with 97 "legal documents related to commercial law", apparently contracts or documents similar to contracts, aiming to identify the sections, subsections, appendices etc. of each document, and to extract the date of each document, the names of the parties involved, the governing law, and jurisdiction. No machine learning was involved. Instead, manually crafted pattern-matching rules were employed.

Deep learning methods have recently been successfully applied to sequence labeling tasks. For example, Ling et al. [38] used a BILSTM layer operating on characters to construct morphology-aware word embeddings, which were combined with WORD2VEC embeddings and passed on to another LSTM or BILSTM layer (with a softmax), to perform language modeling or POS tagging, respectively. Lample et al. [16] experimented with a similar method in named entity recognition, adding a CRF layer, and reporting that an alternative method that involved stacked LSTMs performed worse. Ma and Hovy [17] used a convolutional neural network (CNN) to obtain word embeddings from characters; the word embeddings were subsequently fed to a BILSTM layer followed by a CRF layer to perform POS tagging or named entity recognition. Huang et al. [15] experimented with LSTM and BILSTM layers combined with CRF layers in POS tagging, chunking, and named entity recognition. However, we are among the first to apply deep learning to legal text analytics tasks; see also [39,40].

5. Conclusions and Future Work

Building upon our previous work, we explored how deep learning methods can be used in contract element extraction. We showed that a BILSTM with a logistic regression layer (BILSTM-LR), operating on pre-trained word, POS tag, and token-shape embeddings outperforms in most cases the best methods of our previous work, which employed linear classifiers operating on fixed-size windows of tokens, without employing any manually written rules. Further improvements were observed by stacking an additional LSTM on top of the BILSTM (BILSTM-LSTM-LR) or by adding a CRF layer on top of the BILSTM (BILSTM-CRF). Experimental results indicated that BILSTM-LSTM-LR misclassifies fewer tokens, but that BILSTM-CRF performs better when methods are evaluated for their ability to extract entire contract elements. Interestingly, the additional LSTM and CRF layers were most beneficial in contract element types with few training instances.

Future work could explore if BILSTM-CRF can be improved further by using additional stacked LSTM layers, or additional BILSTM or CNN layers to produce

morphologically-aware word embeddings [38,17]. We also plan to explore data augmentation techniques [41], especially in contract element types with few training instances.

References

[1] Z. Milosevic, S. Gibson, P. F. Linington, J. Cole, and S. Kulkarni. On design and implementation of a contract monitoring facility. In *Proc. of the 1st IEEE Int. Workshop on Electronic Contracting*, pages 62–70, San Diego, CA, 2004. IEEE Computer Society Press.

[2] I. Chalkidis, I. Androutsopoulos, and A. Michos. Extracting contract elements. In *Proc. of the 16th Int. Conf. on Artificial Intelligence and Law*, pages 19–28, London, UK, 2017.

[3] Hsiang-Fu Yu, Fang-Lan Huang, and Chih-Jen Lin. Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning*, 85(1-2):41–75, 2011.

[4] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[5] T. Mikolov, W. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *Proc. of the Conf. of the North American Chapter of the ACL: Human Language Technologies*, pages 746–751, Atlanta, GA, 2013.

[6] J. Pennington, R. Socher, and C. D. Manning. GloVe: Global vectors for word representation. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, pages 1532–1543, Doha, Qatar, 2014.

[7] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[8] Y. Goldberg. *Neural Network Methods in Natural Language Processing*. Morgan and Claypool Publishers, 2017.

[9] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[10] F. A. Gers, J. Schmidhuber, and F. A. Cummins. Learning to forget: Continual prediction with LSTM. *Neural computation*, 12(10):2451–2471, 2000.

[11] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[12] A. Graves, N. Jaitly, and A. Mohamed. Hybrid speech recognition with deep bidirectional LSTM. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278, Olomouc, Czech Republic, 2013.

[13] O. Irsoy and C. Cardie. Deep recursive neural networks for compositionality in language. In *Proc. of the 27th Int. Conf. on Neural Information Processing Systems*, pages 2096–2104, Montreal, Canada, 2014.

[14] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of the 18th Int. Conf. on Machine Learning*, pages 282–289, Williamstown, MA, 2001.

[15] Z. Huang, W. Xu, and K. Yu. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991, 2015.

[16] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proc. of the Conf. of the North American Chapter of the ACL: Human Language Technologies*, pages 260–270, San Diego, California, 2016.

[17] X. Ma and E. Hovy. End-to-end sequence labeling via bi-directional LSTM-cNNs-CRF. In *Proc. of the 54th Annual Meeting of the ACL*, pages 1064–1074, Berlin, Germany, 2016.

[18] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Proc. of the 26th Int. Conf. on Neural Information Processing Systems*, Stateline, NV, 2013.

[19] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber. LSTM: A search space odyssey. *CoRR*, abs/1503.04069, 2015.

[20] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[21] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. of the Int. Conf. on Artificial Intelligence and Statistics*, pages 249–256, Sardinia, Italy, 2010.

[22] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. of the Int. Conf. on Learning Representations*, San Diego, CA, 2015.

[23] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, and G. Kurian. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[24] J. Peng, L. Bo, and J. Xu. Conditional neural fields. In *Advances in Neural Information Processing Systems*, pages 1419–1427. Vancouver, Canada, 2009.

[25] K. Yao, B. Peng, G. Zweig, D. Yu, X. Li, and F. Gao. Recurrent conditional random field for language understanding. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 4077–4081, Florence, Italy, 2014.

[26] D. Andor, C. Alberti, D. Weiss, A. Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. In *Proc. of the 54th Annual Meeting of the ACL (long papers)*, pages 2442–2452, Berlin, Germany, 2016.

[27] M. Curtotti and E. Mccreath. Corpus based classification of text in Australian contracts. In *Proc. of the Australasian Language Technology Association Workshop*, pages 18–26, Melbourne, Australia, 2010.

[28] K. V. Indukuri and P. R. Krishna. Mining e-contract documents to classify clauses. In *Proc. of the 3rd Annual ACM Bangalore Conf.*, pages 7:1–7:5, Bangalore, India, 2010.

[29] X. Gao, M. P. Singh, and P. Mehra. Mining business contracts for service exceptions. *IEEE Transactions on Services Computing*, 5:333–344, 2012.

[30] A. Stranieri and J. Zeleznikow. *Knowledge Discovery from Legal Databases*. Springer, 2005.

[31] C. Biagioli, E. Francesconi, A. Passerini, S. Montemagni, and C. Soria. Automatic semantics extraction in law documents. In *Proc. of the 10th Int. Conf. on Artificial Intelligence and Law*, pages 133–140, Bologna, Italy, 2005.

[32] I. Hasan, J. Parapar, and R. Blanco. Segmentation of legislative documents using a domain-specific lexicon. In *Proc. of the 19th Int. Conf. on Database and Expert Systems Application*, pages 665–669, Turin, Italy, 2008.

[33] E. L. Mencia. Segmentation of legal documents. In *Proc. of the 12th Int. Conf. on Artificial Intelligence and Law*, pages 88–97, Barcelona, Spain, 2009.

[34] E. Francesconi, S. Montemagni, W. Peters, and D. Tiscornia. *Semantic Processing of Legal Texts*. Number 6036 in Lecture Notes in AI. Springer, 2010.

[35] P. Quaresma and T. Goncalves. Using linguistic information and machine learning techniques to identify entities from juridical documents. In E. Francesconi, S. Montemagni, W. Peters, and D. Tiscornia, editors, *Semantic Processing of Legal Texts*, number 6036 in Lecture Notes in AI, pages 44–59. Springer, 2010.

[36] C. Dozier, R. Kondadadi, M. Light, A. Vachher, S. Veeramachaneni, and R. Wudali. Named entity recognition and resolution in legal text. In E. Francesconi, S. Montemagni, W. Peters, and D. Tiscornia, editors, *Semantic Processing of Legal Texts*, number 6036 in Lecture Notes in AI, pages 27–43. Springer, 2010.

[37] M. F. Garcia-Constantino, K. Atkinson, D. Bollegala, K. Chapman, F. Coenen, C. Roberts, and K. Robson. CLIEL: Context-Based Information Extraction from Commercial Law Documents. In *Proc. of the 16th Int. Conf. on Artificial Intelligence and Law*, pages 79–87, London, UK, 2017.

[38] Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. Finding function in form: Compositional character models for open vocabulary word representation. In *Proc. of the Conf. on Empirical Methods in Natural Language Processing*, pages 1520–1530, Lisbon, Portugal, 2015.

[39] J. O' Neill, P. Buitelaar, C. Robin, and L. O' Brien. Classifying Sentential Modality in Legal Language: A Use Case in Financial Regulations, Acts and Directives. In *Proc. of the 16th Int. Conf. on Artificial Intelligence and Law*, pages 159–168, London, UK, 2017.

[40] S. N. Truong, N. Le Minh, K. Satoh, T. Satoshi, and A. Shimazu. Single and multiple layer BI-LSTM-CRF for recognizing requisite and effectuation parts in legal texts. In *Proc. of the 2nd Workshop on Automated Semantic Analysis of Information in Legal Texts*, London, UK, 2017.

[41] T. Devries and G. W. Taylor. Dataset augmentation in feature space. *CoRR*, abs/1702.05538, 2017.