The Utility of Context When Extracting Entities From Legal Documents

Jonathan Donnelly
jonny.donnelly@kirasystems.com
Kira Systems
Toronto, Canada

Adam Roegiest
adam.roegiest@kirasystems.com
Kira Systems
Toronto, Canada

ABSTRACT

When reviewing documents for legal tasks such as Mergers and Acquisitions, granular information (such as start dates and exit clauses) need to be identified and extracted. Inspired by previous work in Named Entity Recognition (NER), we investigate how NER techniques can be leveraged to aid lawyers in this review process. Due to the extremely low prevalence of target information in legal documents, we find that the traditional approach of tagging all sentences in a document is inferior, in both effectiveness and data required to train and predict, to using a first-pass layer to identify sentences that are likely to contain the relevant information and then running the more traditional sentence-level sequence tagging. Moreover, we find that such entity-level models can be improved by training on a balanced sample of relevant and non-relevant sentences. We additionally describe the use of our system in production and how its usage by clients means that deep learning architectures tend to be cost inefficient, especially with respect to the necessary time to train models.

CCS CONCEPTS

• **Information systems** → **Specialized information retrieval**;
• **Computing methodologies** → *Machine learning*.

ACM Reference Format:
Jonathan Donnelly and Adam Roegiest. 2020. The Utility of Context When Extracting Entities From Legal Documents. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3340531.3412746

1 INTRODUCTION

In the practice of Mergers and Acquisitions (M&A) law, lawyers review a company's contracts and associated agreements for potential risk should their client seek to acquire or merge with the other company. During this review, lawyers are tasked with identifying key pieces of information that can help inform decisions related to the transaction (e.g., deal amount). Historically, lawyers would review large numbers of documents manually in short time frames, which could lead to error prone decisions [27]. As the popularity of automated solutions continues to grow [1, 11], we build on previous work [21] to extend automated strategies to identify and extract entities and values.

To aid with our task, we draw motivation from NLP-based sequence tagging [4, 18, 24] where the goal is to label tokens in a body of text. Such labels may include people, places, organizations, products and other textual labels or may correspond to numerical values (e.g., dates, percentages) or even short fragments of text with a known meaning. In general, there is some overlap between the broad categories of traditional sequence tagging and the desired legal information to extract (e.g., dates). On the other hand, lawyers often want only certain information extracted depending on their exact use case. For example, while identifying a location is possible, there may be many such locations in a contract and the lawyer only seeks to identify the jurisdiction in which the contract is legally binding. While previous research [21] has focused on identifying legal information needs in the M&A domain at the sentence level, such coarse-level extractions, as we will see later, return too much extraneous information over and above what is desired.

It is common in sequence tagging tasks that a training instance is simply a sentence interpreted as a sequence of tokens. In such cases, each sentence may have one or more sets of labels being applied. In the case of our task, often only one or two sentences in an entire document contain the desired information. Accordingly, we have a large class imbalance problem, as seen in Table 1, which means that most of our training instances only contain non-relevant labels. To address this issue, we propose a solution that leverages a first-pass sentence layer that uses previously published systems [21] to identify sentences that are likely to contain the relevant information. Once these sentences are identified, we then run what is largely an out-of-the-box sequence tagging layer using Conditional Random Fields (CRFs). While one can alternate how this infrastructure is used for training or for prediction to tailor the efficiency and accuracy of the resulting machine learning models, we show that generally the most training data efficient approach is also superior to other configurations.

This paper presents our experimentation using this two-layer approach. In particular, we vary the amount of training data (all sentences, only relevant sentences, and a balanced sample) and then compare how using a first-pass sentence classifier can improve the effectiveness of the system with respect to the amount of data required and the accuracy of the model with the ultimate finding that the two-layer approach is much more efficient and effective than a single-layer approach. Due to the on-going popularity and success of deep learning methods in NLP, we also evaluate Flair [2], a state-of-the-art deep learning based Named Entity Recognition

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3412746

(NER) solution,[1] which follows the same one layer approach as our baseline. While Flair does achieve effectiveness gains over our baseline solutions, it comes at a much larger computational cost during training and inference. Accordingly, while a deep learning solution, such as Flair, may indeed be the state-of-the-art, it does not appear to be practicable for use in production.

We provide a description of the deployment of this architecture in our production environment over the last several years as well as the limitations that our overall product offering places upon our machine learning. Additionally, we note that the training and testing data used in this paper was generated and curated through the use of our production system by a team of in-house legal professionals with no special machine learning training.[2] We conclude with some thoughts on further directions that our technology might take, especially with regards to accelerating utilization of deep learning architectures.

2 CONTEXT

2.1 Task Description and Constraints

Our document collection consists of a set of over 6,000 documents which have been sourced by a team of legal professionals from the Electronic Data Gathering, Analysis, and Retrieval system (EDGAR)[3] and the System for Electronic Document Analysis and Retrieval (SEDAR)[4] repositories of public filings. From this source collection, the legal professional curates 11 topics, which generally correspond to an entity/value with a specific meaning (e.g., the start date of a contract). For each topic, a subset of the document collection is identified as potentially containing that topic and each document in that subset is annotated for the respective topic.[5] Table 1 shows some statistics on our corpus. In general, we can see that the prevalence of a topic in a document can vary dramatically. In Figure 1, we show an example from our "Governing Law" topic (i.e., the jurisdiction under which a contract is governed). While the gold standard annotation in this example is precisely highlighting a location, a valid solution may annotate more or less information depending on the scope of the topic. In sum, the core "task" our system focuses on addressing is learning a machine learning model from human annotations that matches the human annotations as closely as possible.

To address our task, we have chosen to focus on the identification of an individual topic (binary classification) rather than all possible topics (multi-class classification) simultaneously. We structure the problem in this way due to several restrictions around the user of our platform. In particular, topics may be developed across time due to the cognitive effort of annotating documents for many fields, the potential paucity in examples for a particular topic and differing jurisdictions and languages having different use cases and requirements. Accordingly, should a user need to train a machine learning model to identify a concept, they can do so at their own convenience and as their needs dictate. If we followed more traditional NLP use cases and wanted to train a classifier to identify all instances of all topics then we would impose a high burden on our users. Not only would they need to know all their information needs upfront, they would need to have all their training data available at that instant. To elaborate on how this manifests in our system, we direct you to Section 4.

As we mentioned earlier, lawyers conducting due diligence are often in a time crunch and need to be able to identify information quickly and efficiently. Moreover, they may discover that they need to go back and identify new information as the process evolves. Accordingly, any solution we employ must be quick to train and conduct inference and also be highly accurate. This training efficiency is both in terms of the amount of training data required to achieve high accuracy but also in the associated computational costs. To be more precise let us describe the two primary use cases of our machine learning: pre-trained "out of the box" models and user-defined models.

With pre-trained "out of the box" models we are referring to those machine learning models that are produced by our in-house team of legal professionals. These models attempt to preemptively address the needs of our clients by providing models that are already tailored to their potential use cases. In doing so, the faster that this team can train, validate, and release models due to their high familiarity with the system, we have the potential to spare clients from undertaking this process more than is necessary. This necessity of clients training their own models ties into the second use case where they find that one of our pre-built models does not meet their needs (e.g., due to jurisdictional differences, disagreement about the relevant scope of the model [22]). In these cases, we want to ensure that clients spend as little time training as possible so that they can get down to their core work. In either case, faster and more efficiently enabling the training of high quality models by non-machine learning experts facilitates a better user experience and clients can attain greater value out of the platform.

2.2 Related Work

Prior work in NER has traditionally seen much success by applying sequence models, such as Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs) [7], to this problem of identifying particular classes of entities. This is due in part to the fact that language tends to follow a logical sequence at the sentence level, even if it differs from language to language (e.g., subject-object-verb versus subject-verb-object). We have previously shown [21] that we can exploit similar tendencies in legal documents to tag sequences of sentences (as opposed to sequences of tokens) and that such an approach is superior to binary sentence classification which indicates that the relationships between sentences can be good indicators of relevance.

Due to the explosion of interest in deep learning architectures, new techniques (e.g., BERT [5], Flair [2], ELMo [16]) have pushed boundaries and the the state-of-the-art for sequence tagging tasks on a variety of benchmarks (e.g., CoNLL [28]). Moreover, several NER techniques have leveraged the combined power of CRFs and deep learning models to achieve state-of-the-art results [14]. Due to its recent success in NER, we use the Flair framework [2] as our

[1] Based on the rankings at http://nlpprogress.com/english/named_entity_recognition.html.
[2] Source code for experiments and links to any released data are available from https://github.com/kirasystems/science.
[3] https://www.sec.gov/edgar/search-and-access
[4] https://www.sedar.com/
[5] As we have previously described the exact annotation procedures and behaviours of our users elsewhere [12, 21], we do not reproduce such a description here.

	Avg	Min	Max
Documents	590.08	99	1,895
Documents w/example	485.62	29	1862
Sentences	795,035.69	95,851	2,130,122
Sentences w/example	943.77	46	3131

Table 1: Average, min and max number of documents, documents with examples (*documents* containing a relevant (foreground) entity), number of sentences, and sentences with examples (*sentences* containing a relevant (foreground) entity), across all 11 topics.

representative deep learning baseline in this work. Flair uses pre-trained character level LSTM language models to extract contextual word embeddings. Flair then uses a combination of these contextual embeddings and GloVe embeddings [15] as input to a BiLSTM-CRF sequence classifier. An interesting aspect to glean from Flair is that context is important and that CRFs are still relevant to modern tagging architecture.

2.3 Our task vs "Traditional" NER

Our task and solution differ from those of well studied NER problems, such as the CoNNL 2003 shared task [28], in two ways: first, there is a large data imbalance in our corpora, where we are looking for "needles in a haystack" (i.e., one or two instances per document). Secondly, due to the constraints described above, we prefer a binary classifier for each of our topics, whereas state-of-the-art solutions on public NER datasets (e.g., CoNLL 2003 task [28]) consist of multi-class deep learning models. Due to the resources required to train deep learning models, we argue that they fit into use cases where models are updated infrequently rather than situations requiring rapid iteration. Additionally, the specialized hardware to train deep learning algorithms also means that we place requirements on clients should they decide not to use our platform in the cloud. Such a constraint is not ideal if we want to minimize the cost of maintaining different architectures or allowing clients the ability to control where their data resides. Furthermore, despite state-of-the-art performance on public NER datasets [14], our experiment using Flair [2] in Section 4.4 shows that the performance gain is not large, and is disproportionate to the resources required to attain it.

3 DOCUMENT COLLECTION

To follow the Cranfield Paradigm [3] for our experimental setup, we require a set of topics annotated (or judged) across several topics in a document collection. Due to the nature of the legal profession, it is difficult to acquire legal documents that we can use to make a representative document collection. In the United States and Canada, publicly traded companies are required to produce certain classes of contracts, agreements, and financial documents. Such documents were manually curated from the United States' EDGAR and Canada's SEDAR collections. Our team of legal professionals crawl these repositories for documents that are relevant to their needs and are subsequently included in our document collection. Summary statistics of the collection (post topic creation and annotation) are found in Table 1. We note that until a resulting model is deemed "suitable" to release to customers (i.e., meets internal quality requirements defined by the team) the associated set of training documents may change over time (e.g., often documents are added to reflect particularly nuanced languages).

3.1 Annotation Methodology

Following a similar protocol to that outlined in our previous work [21], our team of lawyers obtained documents relevant to the set of topics and then annotated the regions of text in those documents corresponding to the entities for each topic. The annotation process was conducted using our in-house annotation platform [23] and topics were selected due to perceived benefit to our customers. We worked with our in-house team to select 11 high-quality topics that exhibited the desired phenomenon and were representative of the various use cases that we seek to address for clients. Broadly speaking, these topics correspond to either numerical (e.g., dates, monetary values) or textual (e.g., addresses, locations) values and the titles can be found in Table 3. While end users may wish for extracted values to be normalized in some fashion (e.g., dates normalized into a consistent format), we are only concerned with initial extraction of text which could then be normalized in some later data processing pipeline.

3.2 Data Preprocessing

All documents are initially ran through OCR (due to their nature as PDFs) and annotated using the aforementioned annotation system. Sentence segments are obtained using the "punkt" sentence segmentation algorithm [9] that we trained on a collection of legal documents obtained from EDGAR years ago. Tokenization utilizes a similar set of tuned rules based on the same corpus used to train the sentence segmenter. The fully preprocessed version of our corpus consists of the preprocessed source text for each document along with additional sentence and token labels. Sentence level labels correspond to the presence of an overlap with an annotation (two classes); while token level tags correspond to the position of each token in an annotation or its lack thereof. Following convention, we use BIE tags at the token level, representing the beginning, inside and end of an entity, and a background tag, O (for those tokens outside of any annotation). Figure 1 shows an example of a single instance of the "Governing Law" topic, showing both the sentence and token level tags. As we will discuss in more detail in Section 4, these two levels of tagging are used to train two "levels" of sequence tagger that we can leverage to identify the contextual sentence containing a topic instance and then run a more traditional style sequence tagger on the identified sentence.

4 EXPERIMENTAL METHODOLOGY

As shown in Table 1, each document contains thousands of sentences of which only a few contain the relevant entities. In this section, we describe a baseline single-layer entity model based upon the common NER strategy for entity recognition and a novel two-layer strategy that uses an additional sentence level CRF to first detect contextually relevant sentences and then run a more traditional sequence tagger. The end goal of this two-layer approach is to mitigate the class imbalance issues in our training data. We also discuss the mechanisms for evaluating these strategies at the end of the section.

4.1 One-layer strategy

The one layer strategy is the simplest application of CRFs to our problem as we treat every sentence as a training instance. Figure

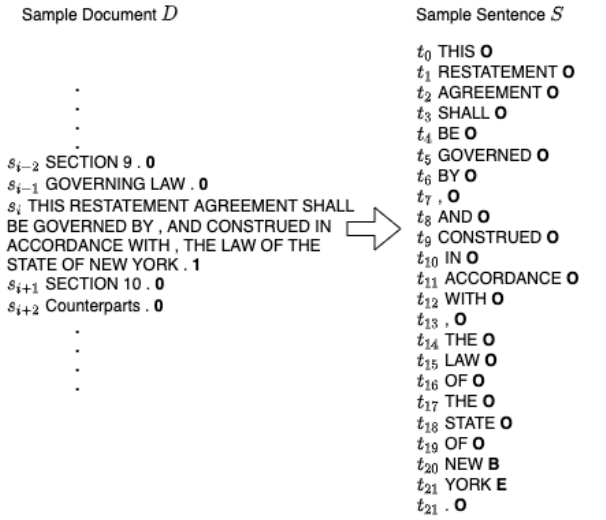


Figure 1: A sample of our training data for the "Governing Law" topic. The left part of the figure shows a sequence of sentences for an example document, where each sentence is labelled to indicate whether it contains (1) or does not contain (0) the desired information. On the right, we show the tokenized version of the relevant sentence with corresponding BIE labels for beginning, inside or end of a topic (foreground) sequence and O for background.

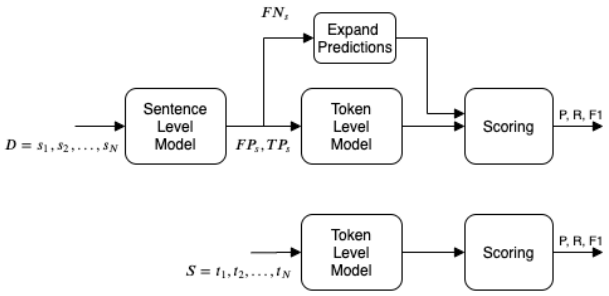


Figure 2: Diagrams illustrating the evaluation of the one-layer (bottom) and two-layer (top) CRF strategies.

2 illustrates the evaluation of this strategy. $S = t_1, t_2, ..., t_N$ represents a sequence of tokens to be tagged, where each token has corresponding n-gram and skip-gram features, which were derived using grid search early into our product's development. To help reduce the memory footprint, we also utilize feature hashing [29] to reduce model complexity. Similar to prior work [21], we examined L-BFGS [10] and Passive-Aggressive (PA) [26] optimization algorithms tuned using grid search, with this paper using L-BFGS with L2-regularization set to 0.1 and maximum iterations to 100. During inference, our model will output BIE token-level tags but, as this work is primarily concerned with identifying the "correct" span of text, we translate all foreground labels (B, I or E) to 1 and all background labels (O) to 0. In other words, we are less concerned with the recall and precision of individual labels and are more focused on measures that represent a user's view of the extracted text.

To tackle the data imbalance issue, we attempt to improve training time by examining three different substrategies using different samples of the training data. In the first substrategy ("1layer-all"), we simply train on all sentences regardless of whether they contain a relevant entity. In the second substrategy ("1layer-pos"), we simply train only on sentences containing a relevant entity to vastly reduce our training set. However, this second strategy has the potential to become too prone to false positives having seen no examples of non-relevant sentences. Accordingly, we use a third substrategy ("1layer-smp") that randomly samples 1 non-relevant sentence for every relevant sentence in a document. The goal of the third strategy is to help balance out the amount of relevant and non-relevant class information. While other ratios are possible, we use this one for simplicity and brevity.

4.2 Two-layer strategy

In our two-layer strategy, we augment the one-layer strategy by training a "sentence-level" CRF that tags whole sentences as potentially containing a relevant entity (i.e., using higher level contextual information). To train the sentence-level model, each document $D = s_1, s_2, ..., s_N$, which is a sequence of sentences, is treated as a training instance. The features used to represent the sentences are a combination of n-gram and skip-gram features as well as word2vec cluster features [21]. To train the sentence-level model, we simply use the defaults of prior work [21]. We then layer the "one layer" entity strategy described above on top of this "sentence-level" model such that any sentences identified by this model are then tagged by the entity model.

Figure 2 illustrates the evaluation of the two-layer strategy. Whole sentences which are tagged as potentially containing an entity, denoted in the diagram as TP_s and FP_s (true positive and false positive sentences, based on the sentence-level labels) are tagged by the token-level model and evaluated as token level extractions. We note that to evaluate this system fairly to the one-layer approach, any false negatives from the sentence model, FN_s, are assigned all background token-level tags for scoring.

The biggest benefit to the two layer approach is that prediction can be substantially faster. Instead of tagging the vast number of sentences we know are likely to be non-relevant, we can isolate sentences that are more likely to be relevant. Accordingly, we can further improve training time depending on how we provide training data to the entity-level CRF and so we have three two-layer sub-strategies that are analogous to those in Section 4.1: "2layer-all," "2layer-pos," and "2layer-smp". We note that 2layer-pos is the closest strategy to the one we deploy in production.

4.3 Evaluation and Metrics

For all experiments we use 5-fold cross validation to evaluate the various strategies. The most straightforward way to produce scores would be to simply count the True Positive, False Positive and False Negatives per token label and calculate Precision and Recall from these. However, one disadvantage of this technique is that longer sequences are given greater weight. Using an example with

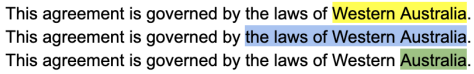


Figure 3: Example of a sentence from corpus with true label and mock predictions. The top line with yellow highlight represents a gold highlight for a jurisdiction. The blue and green highlights represent partially correct predictions.

recall: say our model misses an entity sequence of 5 tokens in length. This will have a greater impact on recall than an entity sequence of 1 token in length since $tp/tp + fn + 5 < tp/tp + fn + 1$. From a user's perspective, it is not necessarily worse to miss longer entity sequences (i.e., any miss is bad). For this reason we choose to consider two other types of measurement.

The first, which we refer to as "Annotation" type Precision, Recall and F1 examines whether the predicted span of text overlaps with the gold standard which is simply a binary overlap measure where any overlap is treated as a success. However, this measure doesn't account for either extraneous or missing segments of overlapping regions. This can be problematic as the sentence-level CRF could produce ideal extractions (predictions spanning entire sentences would overlap with smaller spans within the sentences and extraneous overlap would not be penalized) but would annotate too much text and potentially provide unwanted information to an end user (e.g., all the text around a specific date).[6] To accommodate this, we adopt the plagiarism detection methods of Potthast et al. [17] ("Plagdet") and their macro-averaged Recall, Precision, and F1. These measures should generally account for how much (or little) text overlaps between the predictions and gold standard. Thus, sentence-level annotations would get penalized for highlighting too much extraneous information.

Figure 3 helps illustrate the difference between these two types of scoring with an example of a sentence from our corpus. The true label is highlighted in yellow and examples of foreground predictions on the sentence are highlighted in blue and green. For Annotation type scoring, the blue and green highlights would result in perfect contributions to the scores. However, for Plagdet scoring, the blue highlight would lower precision (too much highlighted) and the green highlight would lower recall (not enough highlighted). When the predicted highlighted section overlaps perfectly with the true highlight, then contributions to both types of scores are the same. The closeness of these two scores provides an indication of how close to perfect our predictions are with respect to the gold standard. Complete false negatives and false positives (not overlapping with a gold annotation at all) affect either evaluation type equivalently.

4.4 Flair experiment

Internally, we have often experimented with deep learning techniques to assess their viability in our production systems but have not seen sufficiently compelling gains to warrant their associated cost. In this section, we provide a case study using a single topic to help illustrate our issues more concretely. To do so, we use Flair [2], described in Section 2.2, that at the time of writing is one of the state-of-the-art solutions for NER that utilizes deep learning [14]. While scientific rigor would require us to run all 11 topics with Flair for a "true" comparison, we also seek to not spend more resources than is necessary to illustrate our point. To that end we evaluate Flair on 1 topic, "Employee Name," that seeks to capture an employee's name from agreements pertaining to their employment. We note that this topic also comprises the fewest documents (and subsequently annotations) of all our topics with the raw text comprising only 80.9Mb of disk space versus the average of 583.2Mb. While we expect all methods to do better with more training data, this method was explicitly chosen to limit the required computational resources as much as possible. In this way, this might not be entirely fair to Flair (i.e., its effectiveness could be better with more data) but we believe that it is sufficient for illustrative purposes. We also note that to ensure we did not erroneously setup Flair, we replicated its performance on the CoNLL 2003 NER task [28].

We ran Flair in exactly the same experimental setup as all other methods which means that we used 5-fold cross validation to evaluate Flair and otherwise train it equivalently to the 1layer-all strategy. We acknowledge that this is not necessarily "status quo" for deep learning techniques but note that such an approach mimics the evaluation that our production has historically used. In an attempt to minimize monetary cost and architectural convenience, we trained Flair using a P4 GPU rather than a more expensive P100 GPU. While this is not an ideal infrastructure, we felt that a P4

Method	Recall	Precision	F1	AnnF1	Time
Flair	80.93	96.26	87.93	90.87	~12 days
1layer-all	71.21	91.42	80.06	83.68	~45 mins
1layer-pos	97.73	1.79	3.51	3.87	~100 seconds
1layer-smp	97.19	18.59	31.21	31.83	~130 seconds
2layer-all	67.77	95.30	79.21	82.28	~50 mins
2layer-pos	79.01	92.59	85.26	86.79	~5.5 mins
2layer-smp	79.25	92.99	85.57	87.03	~6 mins

Table 2: "Plagdet" based Recall, Precision and F1, "Annotation" F1 and approximate time to train and evaluate each method using 5 fold cross validation for our "Employee Name" topic.

Topic \ Strategy	2-lyr-all	2-lyr-pos	2-lyr-smp	1-lyr-all
Employee Name	79.21	85.26	85.57	80.06
Governing Law	87.43	93.75	93.88	84.88
Address of Premises	71.32	82.77	82.89	69.35
Commencement Date	71.51	80.80	80.84	69.98
Rent Amount	35.78	74.10	74.57	40.49
Interest Rate	86.77	91.71	91.71	86.11
Maturity Rate	80.33	86.47	86.47	80.64
Date (UK leases)	63.67	44.98	46.10	63.48
Management Fee	19.82	36.32	36.32	17.78
Duration	74.67	88.14	87.91	68.73
Sq. Ft. of Premises	77.61	87.52	87.57	75.66

Table 3: "Plagdet" based F1 per topic for all the two layer strategies and the 1layer-all strategy. We omit 1layer-pos and 1layer-smp because they are ineffective as standalone solutions without sentence filtering, as can be seen in Figure 4.

[6] It is worth noting that there are edge cases with this type of measurement which must also be accounted for. For example simply highlighting everything as foreground would yield perfect scores.

Method	Recall	Precision	F1	AnnF1
1layer-all	61.9 (21.6)	75.5 (15.0)	67.0 (19.7)	68.1 (19.7)
1layer-pos	97.9 (2.6)	2.6 (2.5)	5.0† (4.7)	5.1 (4.7)
1layer-smp	97.6 (2.6)	19.3 (12.3)	30.6† (16.2)	31.1 (16.4)
Sentence	78.1 (16.9)	17.5 (9.4)	27.4† (12.1)	78.5 (17.3)
2layer-all	58.4 (21.1)	86.4 (12.6)	68.0 (20.3)	69.0 (20.2)
2layer-pos	77.2 (17.6)	81.0 (17.8)	77.4 (18.2)	78.0 (18.0)
2layer-smp	77.1 (17.6)	81.3 (17.6)	77.6 (18.0)	78.2 (17.9)

Table 4: Mean and (standard deviation) Plagdet recall, precision, F1 and Annotation F1 ("AnnF1") across test folds and topics. † denotes that F1 scores for this method are significantly different ($p < 0.01$, paired t-test) to the 2layer-pos strategy.

might more closely resemble the consumer hardware that would be available if someone were to run our system outside of the cloud.

In Table 2, we can very clearly see that Flair, despite the relatively small amount of data, does exceed the effectiveness of all of our approaches, even our in-production architecture. But very critically takes ~2,800 times longer to train, which is roughly 10,000 times the cost (i.e., hundreds of dollars versus fractions of a cent),[7] when compared to that same production architecture. With regard to prediction time, it takes approximately 15 minutes to perform prediction on a single test fold using Flair, which is longer than the time taken to fully evaluate our production architecture using 5 fold cross validation. Accordingly, it becomes very clear that the very slight effectiveness gains made by Flair are more than outdone by the associated cost. Indeed, in fractions of the time, we could very likely conduct a parameter sweep to tune 2layer-pos and achieve equivalent or superior effectiveness in substantially less time. We also readily acknowledge that our Flair solution is not necessarily optimized for production (e.g., determining optimal layer size, number of embeddings, etc), but note that such tuning is also likely to be data dependent and may not be consistent across all potential topics. Accordingly, we find that this baseline Flair implementation, which is identical in configuration to the settings used to achieve state-of-the-art results on the CoNLL 2003 task [28], is reasonable and reflects the general trends we would expect. These results, eliding any infrastructure changes, very strongly disincentivize the use of similar deep architectures in our production environment.

5 RESULTS

When we examine Table 4, we can immediately see that out of the one-layer strategies, using all sentences yields a vastly superior model across topics. This is likely due to the fact that since the one-layer models must predict for every sentence, the sampled and positive-only training regimes means that those strategies overvalue certain features too much. But we note that there is a substantive improvement when just adding one non-relevant example for every relevant exaple, providing some indication that there may be an optimal sampling point at which there is less training data utilized but attains similar performance to 1layer-all.

As we might have expected from Section 4.3, the sentence-level model performs poorly with respect to the "Plagdet" based measures but appears to do seemingly well on annotation-level F1 which reinforces the utility of not just relying on simple binary overlap. In spite of poor sentence-level "Plagdet" performance, the two-layer results shows that this use of contextual information (identifying sentences that are likely to contain the topic of interest) can greatly aid in the performance of the positive-only and sampled training regimes. We posit that this is because this more closely aligns to the training data they received. A similar argument holds with 2layer-all but less dramatically, likely due to the fact that the 1layer-all strategy is reasonably able to distinguish sentence instances that should have a highlight from those that do not due to the copious amount of non-relevant training examples.

An interesting aspect of our evaluation is the near equivalence of plagiarism detection F1 and annotation-level F1 for the one- and two-layer strategies. This likely indicates that when these strategies make a prediction it is generally completely correct (no added or omitted text) and that they are instead just failing to identify some portion of gold annotations or are erroneously identifying entire spans. The small differences show that this isn't always the case but when the predictions align with the gold standard they generally are highly accurate. It is also worth noting that there is a practical upper-bound on the effectiveness of the two-layer strategies, where their recall (at any level) is fundamentally limited by the context sentences identified by the sentence layer.

As might be expected, for some topics, particularly "Management Fee", all strategies do not perform well due to the nature of the annotations and variations around the topics as can be seen in Table 3. Removing this topic generally results in substantial improvements to average effectiveness for all but the 1layer-pos strategy (due to it being consistently bad). Though for the 2layer-pos and 2layer-smp strategies variance is generally halved due to the omission which may provide some indication that these strategies are more sensitive to the training data (annotations for these two topics are less consistent) than other strategies.

In Figure 4, we depict how much training data (in terms of the number of folds) is necessary to achieve saturation for the various strategies with respect to Plagdet F1. It becomes immediately apparent that the 1layer-pos strategy receives no benefit from more data and is a poor performer overall. 1layer-all and the 2layer strategies all have a general increase in performance and reduced variance as more data is added. The biggest improvement comes after the second fold, likely due to further reinforcement of critical features. Overall, 2layer-all is slightly better than 1layer-all indicating that using a contextual filter can be useful in reducing inference time and increasing accuracy with only a small increase in training time.

6 PRODUCTION DEPLOYMENT

The aforementioned 2layer-pos architecture has been deployed in our production environment over the past 6 years with various tweaks to the features used in the intervening years. We note that it is fundamentally an infrastructure that our users themselves (e.g., legal professionals) annotate, train, and assess the quality of the resulting models themselves through a relatively straightforward interface (see Figure 5 for an example).[8] In this way, our users are empowered to train models to extract their desired information

[7] The P100 GPU would reduce this disparity due to shorter training time but not as substantially to make it viable.

[8] We note that we have previously described the user aspects of this interface and behaviours elsewhere [12, 23].

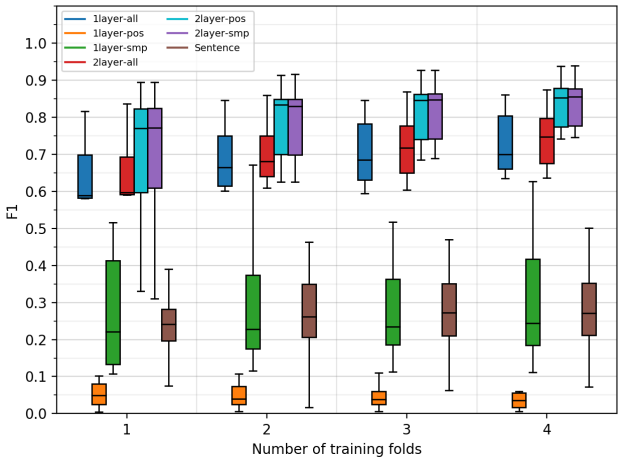


Figure 4: Boxplot illustrating the performance of each of the strategies across all 11 topics as each fold of training data is added. Coloured boxes contain the median and upper and lower quartiles. Whiskers extend to $1.5 \times IQR$ (IQR is the interquartile range) above and below the upper and lower quartiles.

without requiring in-depth machine learning or data science knowledge. While such a "one size fits most" architecture does limit what users can control (e.g., limited hyper-parameter tuning), we have not found this to be particularly troublesome for the vast majority of our users.

This contrast between a more traditional "train once, arbitrary inference," usually by machine learning experts,[9] and "arbitrary training, arbitrary inference" means that our production needs differ from conventional methods. Due to this, we need to be able to "scale up" to meet training demand and ensure models are trained in a timely and cost effective manner. As we saw in Section 4.4, this may not be as feasible as we would like with complex deep learning techniques. While we can achieve an effectiveness improvement using deep learning, it comes at the cost of training time and increased hardware costs and requirements. Essentially, if we want our users to be actively empowered to train models to meet their information needs, we want to reduce the barriers to this training process. Increasing costs and training time is unlikely to facilitate this type of behaviour, especially as our users likely to see progress on model development with relatively few training documents (e.g., 10s to 100s).

While there are always performance optimizations (e.g., fewer embeddings, quantization, pruning) to improve model size and training time, often this comes at the expense of model effectiveness. Performing such optimization may still yield a more costly architecture when contrasted with the relatively low costs of CPU-only training. Even more so when we consider that it may still be cheaper, faster, and more accurate to do per-topic hyper-parameter sweeps with the simpler architecture. Additionally, we have seen that models can be sensitive to the jurisdiction of the documents that they were trained on (e.g., due to differences in law and terminology) and are wary of the burden that this might place upon us to produce "tuned" aspects of the architecture for every possible use case.

The final consideration that is required for our production platform is that some subset of our clients will desire to run the platform on their own hardware.[10] This means that architectural changes of our machine learning have to consider the additional requirements that this can place upon them and potentially on us (e.g., additional support). Moreover, any requirements we set need to be sufficiently justified (e.g., substantial effectiveness increases or training time decreases). We may also allow them to elect not to use such technology but then need to maintain and support the current machine learning architecture that we have described. In either case, we end up having additional costs to meet the needs of these clients and that adds another factor in choosing the right architecture for our platform.

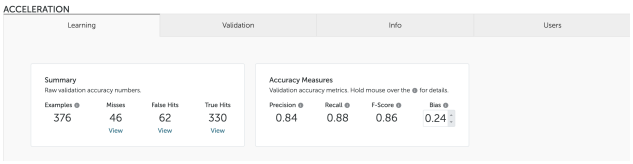
7 FUTURE WORK

While Deep Learning continues to show effectiveness gains in NER and other NLP tasks, these gains come at great computational expense, which we have shown in our experiments. However, Representation Learning is a promising direction for future work. The Flair [2] framework uses a deep learning model as a sequence classifier on top of pre-trained language models and Glove [15] representations. Other works [6, 19] have focused on using linear classifiers on top of representations obtained from pre-trained language models. This approach is advantageous because it requires less energy to train the additional linear model but still requires the use of GPUs to extract the document/sentence/token representation when the language model is large. Beneficially this generation of representations is generally a fixed cost (i.e., often a one time cost) rather than a repeated cost paid every time a model is trained. Other recent research in representation learning has explored how large deep learning architectures [20], different pre-training objectives (e.g., BERT's Masked LM [5]), and different pre-training tasks (e.g., machine translation [25]) result in different representations that can affect downstream performance. In all these cases, the intuition is that the model must produce internal representations of language to perform the pre-training task which may then be useful for other tasks. In our case, another promising, though still computationally heavy approach, is that of "knowledge distillation" [8] in which a smaller model is trained using the predictions of a larger model.

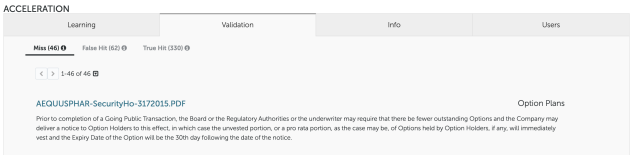
We are also interested in the "explainability" and "interpretability" of NLP models, especially when they are designed for non-experts. Through increased transparency into the how and why of a model, we might be able to facilitate more efficient and effective training processes in our users (i.e., users may have a more concrete "cause and effect" mental model when training than they do now). Moreover, such advances in explainability and interpretability may yield improved feature generation even for simpler models. That is, if deep learning architectures are simply learning more effective variants of our existing features (e.g., n-grams) then we may be able to leverage such insights into equivalently effective shallow

[9] We note that training may occur more than once but is often not done in the production system and once a sufficient quality is achieved is "frozen" until improvements can be made.

[10] This is generally due to contractual obligations around data privacy and governance.



(a) The learning tab allows users to view summary metrics around false positives and negatives. As well, they are able to view more traditional effectiveness measures as calculated using five-fold cross validation. Not pictured is the user's ability to select which data to learn from.



(b) The validation tab allows users to view the missed or extracted text and identify problematic language or consistent trends in behaviour for remediation.

Figure 5: Snapshots of our model training UI that non-experts are able to use to train machine learning models

models.[11] On the other hand, such improvements may also help us identify Niven et al.'s "spurious cues" [13] (e.g., the presence or lack thereof of a potentially unrelated single word). By identifying them, we can then determine whether such cues are worth exploiting or attempting to mitigate. That is, due to the relatively closed domain of our task, it may actually be beneficial to leverage such cues to reduce the required amount of training data. Though such an approach does risk overfitting if not careful.

8 CONCLUSIONS

We have examined how entities can be identified in legal documents in an M&A due diligence context. We have seen that utilizing a traditional NLP sequence tagging model is inferior to using a first-layer to contextualize which sentences might have the desired entity. Moreover, these two layer strategies exceed a single layer in terms of effectiveness with substantially less data. We have also seen that using various sampling strategies can yield much less computational cost but are only truly effective in a two layer approach. To contrast our system architecture, we compare a state-of-the-art deep learning method, Flair [2, 14], which we find to be marginally more effective at a massively increased computational cost.

We also describe how our system architecture is deployed in production and the varying considerations that our less than typical setup of allowing non-experts to train models places upon us. We show that our production setup is such that requiring GPUs and multiple days of training means that our user base would have to wait longer and potentially pay more for models that are only slightly better than what we have now. In summary, our production environment is such that we require highly effective and computationally efficient algorithms lest we impact our users' experience.

REFERENCES

[1] 2017. Legal AI Co.s Seal, Kira + Leverton Show Buoyant Growth. https://www.artificiallawyer.com/2017/09/15/legal-ai-co-s-seal-kira-leverton-show-buoyant-growth/.
[2] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP. In *Proc. ACL '19*.
[3] Cyril W Cleverdon. 1970. *The effect of variations in relevance assessments in comparative experimental tests of index languages.* Technical Report. Cranfield University.
[4] Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 Shared Task on Novel and Emerging Entity Recognition. In *Proc. 3rd Workshop on Noisy User-generated Text*.
[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. NAACL-HLT '19*.

[11] We note that our past work [6] has shown that relying solely on the maximally weighted features does not always yield a good feature in isolation.

[6] Jonathan Donnelly and Adam Roegiest. 2019. On Interpretability and Feature Representations: An Analysis of the Sentiment Neuron. In *Proc. ECIR '19*.
[7] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proc. ACL '05*.
[8] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*.
[9] Tibor Kiss and Jan Strunk. 2006. Unsupervised Multilingual Sentence Boundary Detection. *Computational Linguistics* 32, 4 (2006).
[10] Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Math. Programming* 45, 3, (Ser. B) (1989).
[11] Jeffrey Manns and Robert Anderson. 2017. Engineering Greater Efficiency in Mergers and Acquisitions. 72 (Sept. 2017).
[12] Mary Mikhail, Adam Roegiest, Karen Anello, and Winter Wei. 2020. Dancing with the AI Devil: Investigating the Partnership Between Lawyers and AI. In *CHIIR '20*.
[13] Timothy Niven and Hung-Yu Kao. 2019. Probing Neural Network Comprehension of Natural Language Arguments. In *Proc. ACL '19*.
[14] NLPProgress.com. 2020. NLPProgress.com. http://nlpprogress.com/english/named_entity_recognition.html
[15] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.
[16] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proc. NAACL '18*.
[17] Martin Potthast, Benno Stein, Alberto Barrón-Cedeño, and Paolo Rosso. 2010. An Evaluation Framework for Plagiarism Detection. In *Proc. COLING '10*.
[18] Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards Robust Linguistic Analysis using OntoNotes. In *Proc. CoNLL '13*.
[19] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2018. *Learning To Generate Reviews and Discovering Sentiment.* Technical Report. OpenAI.
[20] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. *Language Models are Unsupervised Multitask Learners.* Technical Report. OpenAI.
[21] Adam Roegiest, Alexander K. Hudek, and Anne McNulty. 2018. A Dataset and an Examination of Identifying Passages for Due Diligence.. In *Proc. SIGIR '18*.
[22] Adam Roegiest and Anne McNulty. 2019. Variations in Assessor Agreement in Due Diligence. In *Proc. CHIIR '19*.
[23] Adam Roegiest and Winter Wei. 2018. Redesigning Document Viewer for Legal Documents. In *Proc. CHIIR '18*.
[24] Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. *CoRR* cs.CL/0306050 (2003).
[25] Holger Schwenk and Matthijs Douze. 2017. Learning Joint Multilingual Sentence Representations with Neural Machine Translation. In *Proc. 2nd Workshop on Representation Learning for NLP*.
[26] Shai Shalev-shwartz, Koby Crammer, Ofer Dekel, and Yoram Singer. [n.d.]. Online Passive-Aggressive Algorithms. In *Proc. NeurIPS 2004*.
[27] Debbie Stephenson. 2013. Top 10 Due Diligence Disasters. https://www.firmex.com/thedealroom/top-10-due-diligence-disasters/.
[28] Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proc. NAACL-HLT '03*.
[29] Kilian Q. Weinberger, Anirban Dasgupta, Josh Attenberg, John Langford, and Alexander J. Smola. 2009. Feature Hashing for Large Scale Multitask Learning. *CoRR* abs/0902.2206 (2009).