Deep Ensemble Learning for Legal Query Understanding

Arunprasath Shankar
LexisNexis
Raleigh, USA
arunprasath.shankar@lexisnexis.com

Venkata Nagaraju Buddarapu
LexisNexis
Raleigh, USA
venkatanagaraju.buddarapu@lexisnexis.com

ABSTRACT

Legal query understanding is a complex problem that involves two natural language processing (NLP) tasks that needs to be solved together: (i) identifying intent of the user and (ii) recognizing entities within the queries. The problem equates to decomposing a legal query into its individual components and deciphering the underlying differences that can occur due to pragmatics. Identifying the desired intent and recognizing correct entities helps us return back relevant results to the user. Deep Neural Networks (DNNs) have recently achieved great success surpassing traditional statistical approaches. In this work, we experiment with several DNN architectures towards legal query intent classification and entity recognition. Deep Neural architectures like Recurrent Neural Networks (RNNs), Long Short Term Memory (LSTM), Convolutional Neural Networks (CNNs) and Gated Recurrent Units (GRU) were applied and compared against one another both individually and as combinations. The models were also compared against machine learning (ML) and rule-based approaches. In this paper, we describe a methodology that integrates posterior probabilities produced by the best DNN models and create a stacked framework for combining the different predictors to improve prediction accuracy and F-measure for legal intent classification and entity recognition.

CCS CONCEPTS

• **Information systems → Query intent**; *Query representation*; *Query log analysis*; • **Computing methodologies → Information extraction**; **Neural networks**; **Ensemble methods**;

KEYWORDS

Deep Learning, Recurrent Neural Networks, Convolutional Neural Networks, Long Short Term Memory, Ensemble Learning, Query Intent, Named Entity Recognition, Legal

1. INTRODUCTION

With US legal services market approaching $500 bn in 2018 that is ≈2% of US GDP [1] and substantial incentives to increase market share, maximizing user satisfaction with search results continues to be the primary focus for legal search industry. Recently, there is an upward trend of embracing vertical search engines by legal companies since they provide a specialized type of information service to satisfy a user's intent. Using a specialized user interface, a vertical search engine can return more relevant results than a general search engine for in-domain legal queries.

Copyright © 2018 for this paper held by its authors. Copying permitted for private and academic purposes.

In practice, a user has to decide his choice of vertical search engine beforehand to satisfy his query intent. It would be convenient if a query intent identifier could be provided in a general search engine that could precisely predict whether a query should trigger a vertical search in a certain domain. Moreover, since a user's query may implicitly express more than one intent, it would be very helpful if a general search engine could detect all query intents, distribute it to appropriate vertical search engines and effectively organize the results from the different vertical search engines to satisfy a user's need. Consequently, understanding a query intent is crucial for providing better search results and thus improving the overall satisfaction of the user.

Understanding intent and identifying legal entities from a user's query can help legal search automatically route the query to corresponding vertical search engines and obtain relevant contents, thus, greatly improving user satisfaction meaning better assistance to law researches to support legal arguments and decisions. Law researchers have to cope with a tremendous load of legal content since sources of law can originate from diversified sources like judicial branch, legislative branch (statutes), legal reference books, journals, news etc. This makes legal search and understanding queries in legal search such an important aspect to retrieve relevant support documents for supporting one's argument.

Legal search is hard as it demands writing complex queries to retrieve desired content from information retrieval (IR) systems. Classifying legal queries and identifying domain specific legal entities from queries is even harder. For e.g., in the query: "*who is supreme court magistrate John Roberts and abortion law?*", the word "*magistrate*" can be resolved to a <judge title> when observed along with the context phrase "*supreme court*". Similarly, the phrase "*abortion law*" can be identified as a <practice area> when seen alongside a supporting context. However, since we also observe the interrogative phrase "*who is*", we can safely assume that the intent of this query is <judge> or <person> search.

Similar to google queries, legal queries can be classified into one of three categories. (i) **Do**: The users wants to do something, like buy a product or service. E.g., Buy Law Books, Research Guides, Police/Personal Reports, Real Estate Property Records etc. (ii) **Know**: An informational query, where the user wants to learn about a subject. E.g., law, statute, doctrine etc. Very often single word queries are classified at least partially as "Know" queries. (iii) **Go**: Also known as a navigational query, the user wants to go to a specific site scoped to a particular legal entity. E.g., a query where a user wants to know analytics of a specific legal entity like a judge or an expert witness. Our research in this work focuses only on "Go" type of queries, e.g., an user wanting to understand a particular judge profile with the query "*judge John D. Roberts*".

The process of finding named entities in a text and classifying them to a semantic type is called named entity recognition (NER). Legal NER is nearly always used in conjunction with intent classification systems. Given a query, the goal of NER system described in this paper is two-fold: (i) segment the input into semantic chunks, and (ii) classify each chunk into a predefined set of semantic classes. For e.g., given a query "judge john d roberts", the desired output would be: "*judge*" = <OTHER> and "*john d roberts*" = <JUDGE>. Here the class <JUDGE> represents a person or a judge entity and the class <OTHER> represents any non judge specific term.

The aim of this research is to improve the understanding of queries involved in legal research. In this paper, we explore three different approaches: (a) machine learning (ML) with feature engineering, (b). deep learning (DL) without any feature engineering and (c). ensemble of deep neural networks. We then perform quantitative evaluations in comparison to an already existing baseline model which is a rule-based classification system in production. Finally, we show from our experimental results that a deep ensemble model significantly outperforms other approaches for both intent classification and legal NER.

2. BACKGROUND AND RELATED WORK

DL systems have dramatically improved the state of several domains like NLP, computer vision, image processing etc. Various deep architectures and learning methods have been developed with distinct strengths and weaknesses in recent years. Deep ensemble learning is a learning paradigm where ensembles of several neural networks show improved generalization capabilities that outperform those of single networks. For deep learning of multi-layer neural networks, ensemble learning is still applicable [2, 3]. However, there is not much work done towards legal domain involving neither deep learning nor ensemble learning. How can ensemble learning be applied to various DNN architectures to achieve better results for legal tasks is the primary focus of this paper.

Most ML approaches to text understanding consists of tokenizing a string of characters into structures such as words, phrases, sentences or paragraphs, and then apply some statistical classification algorithm onto the statistics of such structures [4]. These techniques work well when applied to a narrowly defined domain.

Typical queries submitted to legal search engines contain very short keyword phrases, which are generally insufficient to fully describe a user's information need. Thus, it is a challenging problem to classify millions of queries into some predefined categories. A variety of related topical query classification problems have been investigated in the past [5] [6]. Most of them seek to use statistical machine learning methods to train a classifier to predict the category of an input query. From the statistical learning perspective, in order to obtain a classifier that has good generalization ability in predicting future unseen data, two conditions should be satisfied: discriminative feature representation and sufficient training samples. However, for the problem of query intent classification, even though there are huge volumes of legal queries, both conditions are hardly to met due to the sparseness of query features coupled with the sparseness of labeled training data. In [5], Beitzel et al. attempted to solve this problem through augmenting the query with more features using external knowledge, such as search engine results and achieved fair results.

DNNs have revolutionized the field of NLP. RNNs and CNNs, the two main types of DNN architectures are widely explored to handle various NLP tasks. CNN is supposed to be good at extracting positional invariant features and RNN at modeling units in sequence. The state-of-the-art on many NLP tasks often switches due to the battle of CNNs and RNNs. While CNNs take advantage of local coherence in the input to cut down on the number of weights, RNNs are used to process sequential data (often with LSTM cells). RNNs are also good at representing very large implicit internal structures that are difficult even to think about.

In summary, the conventional wisdom is that RNNs should be used when the context is richer and there is more state information that needs to be captured. This proposition has been challenged by CNNs recently with the claim that finite state information of limited scope can be more efficiently handled by multiple convolution layers. We think both are true, and one should not go for RNNs just for the sake of it, instead more efficient deep CNNs should be tried in limited context situations. But for more complex implicit mappings where context and state information spans are much bigger, RNNs are the best and at this point almost the only tool.

There is not a lot of previous research work involving DL and legal domain for the problems of intent classification and NER. However, there are a handful of papers that talk about DNN approaches for non-legal intent classification and general NER. Also recently, RNNs and CNNs have been applied on a variety of NLP tasks with various degree of success. Below, we talk about the evolution of various DNNs and their applications towards NLP tasks.

In [7], Zhai et al. adopted RNNs as building blocks to learn desired representations from massive user click logs. The authors proposed a novel attention network that learns to assign attention scores to words within a sequence (query or ad). In [8], Hochreiter and Schmidhuber introduced LSTM which solved the most complex, artificial long-time-lag tasks that have never been solved by previous recurrent network algorithms. In [9], Chung et al. compared different types of recurrent units in RNNs especially focusing on units that implement a gating mechanism, such as LSTM and GRU units. They evaluated these units on the tasks of polyphonic music modeling and speech signal modeling and proved LSTMs and GRUs work better than traditional recurrent units.

In [10], Hinton et al. introduced CNN and used it to for image classification on the ImageNet dataset and established new state-of-the-art results. In [11], LeCun et al. demonstrated that we can apply DL to text understanding from character-level inputs all the way up to abstract text concepts, using temporal CNNs. They applied CNNs to various large-scale datasets, including ontology classification, sentiment analysis, text categorization etc. and showed that temporal CNNs can achieve astonishing performance without any prior knowledge of syntactic or semantic structures with regards to a human language. With respect to intent classification, Hu et al. devised a methodology to identify query intent by mapping the query to a representation space backed by Wikipedia [12]. In [13], the authors applied CNNs for intent classification and achieved results in par with state-of-the-results.

There are also many recent works that combine RNNs with CNNs for different NLP tasks. For example, in [14], Chiu et al. presented a novel neural network architecture that automatically detects word and character-level features using a hybrid bidirectional LSTM and CNN architecture, eliminating the need for most of feature engineering and established a new state-of-the-art performance with an F1 score of 91.62% on CoNLL-2003 data set. In [15], Limsopatham et al. proposed a bidirectional LSTM (Bi-LSTM) to automatically learn orthographic features from tweets.

There is a handful of papers that delve into legal applications using DNNs. In [16], Sugathadasa et al. proposed a system that includes a page ranking graph network with TF-IDF to build document embeddings by creating a vector space for the legal domain, which can be trained using a doc2vec neural network model supporting incremental and extensive training for scalability of the system. In [17], Nanda et al. proposed a hybrid model using LSTM and CNN which utilizes word embeddings trained on the Google News vectors and evaluated the results on COLIEE 2017 dataset. They demonstrated that the performance of LSTM + CNN model was competitive with other textual entailment systems. Similarly, in [18], the authors proposed a methodology to employ DNNs and word2vec for retrieval of civil articles.

For NER, Huang et al. proposed a variety of LSTM and LSTM variants like LSTM + CRF for NER and achieved state-of-the-art results [19]. Recently this year, in [20], Peters et al. introduced a new type of deep contextualized word representation that models both semantics and polysemy and improved the state of the art across six challenging NLP problems, including question answering, textual entailment, sentiment analysis and NER.

In this paper, we scope our research limited to judge queries meaning queries that are meant to be a search for judge profile. Being this the scope of the problem, the intent classification task needs to classify a given query as a judge query or not. Once the query is identified as a judge query, the NER system that follows, recognizes if the query contains any person entities. The entities are then routed to vertical searches that follow. The structure of the paper is as follows. Section 3. discusses the various experiments that was carried out to solve legal query understanding. That is followed by Section 4. that presents and analyses overall results. The paper ends with Section 5. which gives the conclusion and discussion on future work.

3. MODELS AND EXPERIMENTS

The adoption of artificial Intelligence (AI) technology is undoubtedly transforming the practice of law. Many in the legal profession are aware that using AI can greatly reduce time and costs while increasing prediction measures. In the following subsections, we talk about data collection, augmentation and training of the different DNN models for legal tasks we experimented for: (i) intent classification and (ii) legal entity recognition.

3.1 Data Collection

For the purpose of our experiments, we created data sets comprising of different types of legal queries such as judge search, case law search, statutes/elements search, etc. For intent classification, since the classification task is binary, we labeled all judge queries to be positive and the rest as negative. For NER, we used three labels: (i) <OTHER> used to tag any tokens that are not part of a judge name (ii) <B-PER> denotes the beginning of a person (judge) name and (iii) <I-PER> denotes the inside of a name. Table 1 portrays the different query types by volume we used for our experiments. All of the collected data labeled by our subject matter experts (SMEs). All of the data discussed here are proprietary to LexisNexis.

3.2 Data Augmentation and Balancing

The labeled data was good enough to get started but was highly imbalanced with respect to NER labels. In order to balance out the data set, we augmented the data by (i) expanding queries by pattern using judge names from our proprietary judge master database, (ii) oversampling the under represented patterns and (iii) under sampled the over exemplified patterns. The top 10 patterns by frequency are shown in table 3. These 10 patterns alone contributed to ≈78% of the labeled data. The balancing act was carried out by following two strategies: For oversampling, we pick a pattern and create synthetic data points (queries) by randomly substituting judge names/titles around the patterns to be fitted. For the process of under sampling, random data points are selected from buckets of pattern and removed. The sampling process is stopped when there are equal number of queries in all the buckets of patterns. Once the data is augmented and balanced, we split the data into train, dev and test sets. Table 2 shows the ratio of data split for both intent classification and NER.

Type	Example	Volume %
Judge	judge John D. Roberts	34
Word Wheel	Ohio Municipal Court, Bellefontaine	6
Query Log	sexual harassment	5
Statute	statute /s limitations /s actual /s fraud	31
Elements Law	contract defense unconscionability elements	20
Case Search	Powers v. USAA	4

Table 1: Data Types by Volume

	Intent	NER
Train	505,760	1,052,000
Dev	20,000	20,000
Test	20,000	20,000

Table 2: Data Sets

Pattern	Count
B-PER I-PER I-PER	217,374
B-PER I-PER	109,500
O B-PER I-PER I-PER	92,436
B-PER I-PER I-PER I-PER	73,668
O B-PER I-PER	53,896
O B-PER I-PER I-PER I-PER I-PER	48,157
B-PER I-PER I-PER I-PER	39,970
B-PER I-PER I-PER I-PER I-PER	33,474
O B-PER	22,986
O O O B-PER I-PER I-PER I-PER O	9,968

Table 3: Top 10 Patterns

3.3 Word Embedding

Learning a high-dimensional dense representation for vocabulary terms, also known as a word embedding, has recently attracted much attention in NLP and IR tasks. The embedding vectors are typically learned based on term proximity in a large corpus and is used to accurately predict adjacent word(s) for a given word or context [21]. For the purpose of training our NER DNN models, we created word vectors of size 580,614 to be used as word embeddings to the embedding layer of our DNN models. The word vectors were created by training a word2vec continous bag of words (CBOW) model on AWS ml.p3.2xlarge instance with a single NVIDIA Tesla V100 GPU. As an input to the model, 10 Million queries were obtained from user session logs were used. These queries were ordered by frequency. The word2vec model was trained in 112.2 minutes for 100 epochs.

3.4 Intent Classification

3.4.1 Baseline vs ML Approaches: For intent classification, we tried a few different ML classifiers firsthand before delving into the deep learning arena. The ML models were compared against a rule-based query recognition system highlighted ● that was already established as our baseline system to improve. Classifiers were selected from both the linear and non-linear category. The list of classifiers tried and their corresponding results (F1 score) against baseline are shown in table 4 below. Our main requirement while picking and implementing the classifiers was to minimize the overall number of false positives. This is because false positives with respect to intent classification can intercept queries that are meant to be routed to a different vertical search engine affecting the overall customer satisfaction towards the product. Amongst the linear classifiers, linear SVM seemed to perform slightly better than the logistic regression classifier. Amongst the non-linear category, the multi-layer perceptron highlighted ● seemed to perform the best beating decision trees, adaboost and naive bayes approaches.

3.4.2 Feature Engineering for ML Classifiers: There are different ways one can address a judge in legal taxonomy such as “chief justice”, “associate justice”, “magistrate” etc. All of the different phrases used for addressing a judge are compiled into a bag of words representation. In addition to bag of words, all the ML classifiers use POS, gazetteer, word shape and orthographic features to represent semantic and linguistic meaning of a query.

	Dev			Test		
Model	F	FP	FN	F	FP	FN
Rule Engine (Baseline)	98.66	57	209	98.69	52	208
Logistic Regression	98.31	137	141	98.48	132	120
Gaussian NB	89.79	1789	20	90.08	1749	20
AdaBoost	97.58	137	256	97.54	165	241
Decision Tree	95.75	25	647	95.96	25	623
Linear SVM	98.28	139	142	98.54	124	117
Multi-layer Perceptron	98.36	107	160	98.57	119	117

Table 4: Intent Classification - Baseline vs ML

3.4.3 LSTM based Intent Classifier: The LSTM, first described in [8], attempts to circumvent the vanishing gradient problem by separating the memory and output representation, and having each dimension of the current memory unit depending linearly on the memory unit of the previous time step. The DNN architecture we tried using LSTM is shown in figure 1. The embedding layer that feeds the LSTM layer is composed of a vocabulary of 49,701 words with an output dimension of 200. 100 hidden units are used for the LSTM layer. The flatten layer uses 200 hidden units. The flatten layer takes a tensor of any shape and transform it into a one dimensional tensor. Both uni and bi-directional LSTMs were trained for the task.

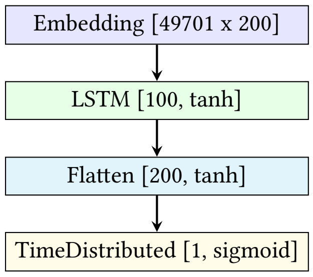

Figure 1: LSTM for Intent Classification

3.4.4 CNN based Intent Classifier: CNNs are built out of many layers of pattern recognizers stacked on top of each other. Convolutional is a way of saying that the machine looks at small parts of a query first rather than trying to account for the whole thing. Each successive layer combines information from these small parts to fill in the bigger picture and assemble complex patterns of meaning. After trying few variants of LSTM architectures, we started experimenting with CNNs for intent classification. The general neural architecture we used for CNN is show in figure 2. The major difference here compared to the LSTM models are: CNNs use two dense layers after the flatten layer whereas LSTMs use only one layer. The 1D convolutional layer uses 32 filters with a kernel size = 8 and the max pooling layer uses 2 strides with a pool size = 2.

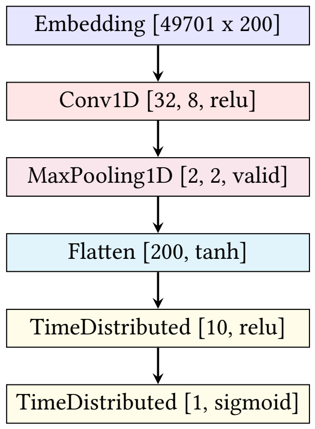
Embedding [49701 x 200]
Conv1D [32, 8, relu]
MaxPooling1D [2, 2, valid]
Flatten [200, tanh]
TimeDistributed [10, relu]
TimeDistributed [1, sigmoid]

Figure 2: CNN for Intent Classification

3.4.5 Hybrid Models: As a next step, the best performing models from both the LSTM and CNN pools were picked and combined into a hybrid model for classification. We built two models for this experiment, LSTM + CNN and Bi-LSTM + CNN. The architectural diagram for Bi-LSTM + CNN model is shown in figure 3.

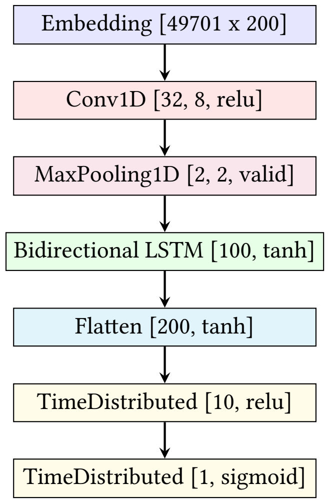
Embedding [49701 x 200]
Conv1D [32, 8, relu]
MaxPooling1D [2, 2, valid]
Bidirectional LSTM [100, tanh]
Flatten [200, tanh]
TimeDistributed [10, relu]
TimeDistributed [1, sigmoid]

Figure 3: Bi-LSTM + CNN for Intent Classification

3.4.6 Deep Ensemble for Intent Classification: Ensemble learning is a ML paradigm where multiple learners are trained to solve the same problem. In contrast to ordinary ML approaches which try to learn one hypothesis from training data, ensemble methods try to construct a set of hypotheses and combine them to use. For intent classification, we use stacking which applies several models to original data. In stacking we don't have just an empirical formula for our weight function, rather use a logistic regression model to estimate the input together with outputs of every model to estimate the weights or, in other words, to determine what models perform well and what badly given these input data.

To simplify the stacking procedure for ensemble learning, we perform a linear combination of the original class posterior probabilities produced by the best DNN models at the word level (see table 8). A set of parameters in the form of full matrices are associated with the linear combination, which are learned using the training data consisting of the word-level posterior probabilities of the different models and its corresponding word-level target values (0 or 1). Figure 4 depicts the model architecture of the ensemble classifier for the task of intent classification.

3.4.7 Training: Table 5 below shows the overall training statistics for the different DNN models deployed for intent classification. The models were trained on AWS ml.p3.8xlarge instance with 4 NVIDIA Tesla V100 GPUs. Average time shown below is measured in minutes.

	Avg Time	**Epochs**	**Batch Size**
LSTM	75.54	50	1000
Bi-LSTM	93.72	30	1000
CNN	37.22	100	500
LSTM + CNN	84.21	50	1000
Bi-LSTM + CNN	101.45	50	1000

Table 5: Train Statistics - Intent Classification

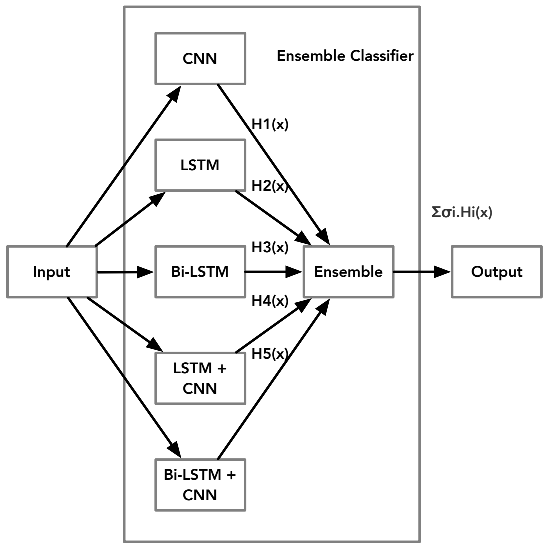

Figure 4: Ensemble Classifier for Intent Classification

3.5 Named Entity Recognition

3.5.1 Baseline: Baseline model for NER is shown in table 12 . The baseline model (highlighted in ●) was previously established and it is the same rule-based system that was set as baseline for intent classification. The row highlighted in ● shows metrics from a conditional random field (CRF) probabilistic model that was previously implemented. This model was eventually discarded since it was outperformed by most of DNN models.

	Architecture			Dev			Test		
	Dropout	Hidden Units	Embedding	F-score	FP	FN	F-score	FP	FN
LSTM I	False	50	200	99.74	25	27	99.72	14	42
LSTM II	False	50	100	99.64	52	20	99.74	26	27
LSTM III	False	50	200	99.77	23	24	99.79	17	25
LSTM IV	True	50	200	99.66	54	15	99.78	24	21
Bi-LSTM I	False	100	200	99.83	19	20	99.84	11	22
Bi-LSTM II	True	100	200	99.72	38	18	99.76	26	23

Table 6: Intent Classification - LSTM

	Architecture			Dev			Test		
	Filters	Kernel	Padding	F-score	FP	FN	F-score	FP	FN
CNN I	32	16	same	99.83	21	13	99.83	15	18
CNN II	32	8	same	99.84	19	13	99.83	14	18
CNN III	64	8	same	99.82	24	11	99.84	13	18
CNN IV	64	16	same	99.78	28	15	99.89	15	15
CNN V	64	4	same	99.81	31	9	99.83	20	14

Table 7: Intent Classification - CNN

	Dev			Test		
Model	F-score	FP	FN	F	FP	FN
LSTM III	99.77	23	24	99.79	17	25
Bi-LSTM I	99.81	19	20	99.84	11	22
CNN II	99.84	19	13	99.83	14	18
LSTM III + CNN II	99.81	26	14	99.90	10	11
Bi-LSTM I + CNN II	99.88	14	11	99.86	7	22
Ensemble (Top 5)	99.91	9	9	99.91	4	15

Table 8: Intent Classification - Winners & Ensemble

	Architecture			Dev			Test		
	Filters	Kernel	Padding	Precision	Recall	F-score	Precision	Recall	F-score
CNN I	128	5	same	99.2707	99.2654	99.2665	99.0583	99.0511	99.0526
CNN II	128	5	same	98.6557	98.6435	98.6468	98.9799	98.9654	98.9681
CNN III	64	5	same	99.2632	99.2549	99.2564	99.0729	99.0605	99.0627
CNN IV	128	10	same	99.3972	99.3967	99.3969	99.2193	99.2177	99.2181

Table 9: Named Entity Recognition - CNN

3.5.2 RNN based Named Entity Recognition: For NER, we started with RNNs. The neural architecture for RNN is shown in figure 5 below. For the embedding layer, we used pre-trained word2vec embeddings of dimensions (577,149 x 200). For the RNN layer, 200 hidden units were used. The time distributed output layer has 3 units and uses a softmax function since we have 3 classes in total (<OTHER>, <B-PER> & <I-PER>. A dropout value of 0.2 was also used for the configuration.

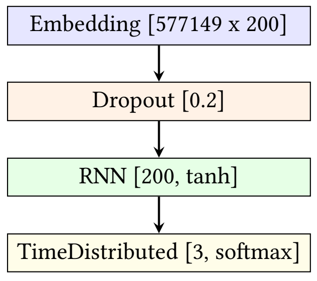

Figure 5: RNN for NER

	Architecture			Dev			Test		
	Dropout	Hidden Units	Pad Length	Precision	Recall	F-score	Precision	Recall	F-score
RNN I	0.4	100	20	99.1119	99.1022	99.1042	98.8990	98.8876	98.8900
RNN II	0.4	100	20	99.1171	99.1063	99.1084	98.9171	98.9039	98.9065
RNN III	0.2	100	20	99.1750	99.1665	99.1682	98.9818	98.9711	98.9733
LSTM	0.4	100	20	99.1727	99.1624	99.1643	98.9894	98.9764	98.9788
Bi-LSTM	0.4	100	20	99.5344	99.5331	99.5334	99.3422	99.3397	99.3403
GRU	0.4	100	20	99.1256	99.1227	99.1235	98.8564	98.8523	98.8534
Bi-GRU	0.4	100	20	99.4734	99.4733	99.4734	99.2931	99.2929	99.293

Table 10: Named Entity Recognition - Recurrent Neural Networks

	Dev			Test		
Model	Precision	Recall	F-score	Precision	Recall	F-score
RNN III + CNN IV	99.3812	99.3794	99.3798	99.2027	99.1998	99.2635
LSTM + CNN IV	99.4036	99.3995	99.4002	99.269	99.2624	99.2635
Bi-LSTM + CNN IV	99.5286	99.5262	99.5267	99.4043	99.4007	99.4013
GRU + CNN IV	99.4365	99.4323	99.4330	99.2528	99.2466	99.2477
Bi-GRU + CNN IV	99.532	99.5294	99.5299	99.3829	99.3786	99.3793

Table 11: Named Entity Recognition - Hybrid Models

	Dev			Test		
Model	Precision	Recall	F-score	Precision	Recall	F-score
Rule Engine (Baseline)	93.2794	92.9001	92.7009	94.0157	93.7872	93.6778
CRF	92.1442	89.3441	91.3463	91.3421	90.0323	90.6341
CNN IV	99.3972	99.3967	99.3969	99.2193	99.2177	99.2181
LSTM + CNN IV	99.4036	99.3995	99.4002	99.2692	99.2624	99.2635
Bi-LSTM	99.5344	99.5331	99.5334	99.3422	99.3397	99.3403
Bi-LSTM + CNN IV	99.5286	99.5262	99.5267	99.4043	99.4007	99.4013
Bi-GRU + CNN IV	99.5323	99.5294	99.5299	99.3829	99.3786	99.3793
Ensemble (Top 5)	99.5596	99.5577	99.5581	99.4193	99.4159	99.4165

Table 12: Named Entity Recognition - Winners & Ensemble

3.5.3 LSTM based Named Entity Recognition: LSTMs in general circumvent the vanishing gradient problem faced by RNNs. For NER, we used both uni and bi-directional LSTMs. The architecture is similar to shown in figure 5 except the RNN layer is replaced by a LSTM layer.

3.5.4 CNN based Named Entity Recognition: For NER, we also experimented with CNN. The neural architecture for CNN is shown in figure 6. The Conv1D layer consists of 128 filters with kernel size set to 5. In contrary to the CNN used for intent classification, this architecture does not use a max pooling layer.

3.5.5 GRU based Named Entity Recognition: More recently, gated recurrent units have been proposed [9] as a simplification of the LSTM, while keeping the ability to retain information over long sequences. Unlike LSTM, GRU uses only two gates, memory units do not exist, and the linear interpolation occurs in the hidden state. As part of our experiment, we replaced the RNN layer in the architecture shown in figure 5 with a GRU layer.

3.5.6 Hybrid Models: For NER, we combined the above discussed models. Some of the hybrid models we built are RNN + CNN, LSTM + CNN and GRU + CNN (both uni and bi-directional). Architecture of Bi-LSTM + CNN is shown in figure 7.

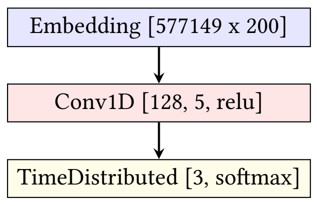

Figure 6: CNN for NER

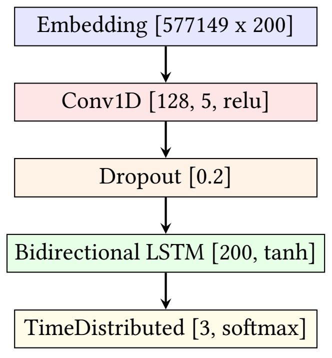

Figure 7: Bi-LSTM + CNN for NER

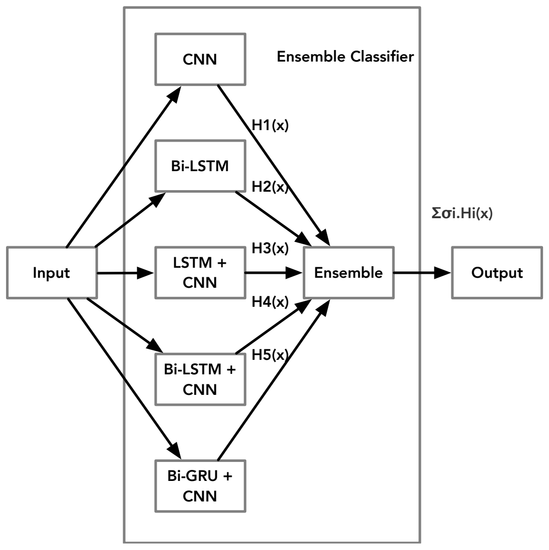

Figure 8: Ensemble Classifier for Named Entity Recognition

3.5.7 Deep Ensemble for Named Entity Recognition: To create an ensemble for NER, the DNN models are ranked by their F1 score. The top 5 best models are then picked and stacked into an ensemble. Ensemble of top 10 models was also experimented and discarded since it underperformed compared to the ensemble of top 5. Figure 8 shows the architecture of the chosen ensemble model.

3.5.8 Training: The training time for epochs are show in table 13 below. A batch here corresponds to a chunk of user input queries. The neural architectures were implemented using tensorflow, keras and scikit-learn.

	Avg Time	**Epochs**	**Batch Size**
RNN	85.23	50	1000
LSTM	104.38	50	1000
Bi-LSTM	123.84	50	1000
GRU	94.2	50	1000
Bi-GRU	104.27	50	1000
CNN	72.02	100	500
RNN + CNN	97.93	50	1000
LSTM + CNN	134.81	50	1000
Bi-LSTM + CNN	154.43	50	1000
GRU + CNN	115.24	50	1000
Bi-GRU + CNN	132.64	50	1000

Table 13: Train Statistics - Named Entity Recognition

4. RESULTS

4.1 Evaluation Metrics

We utilize standard measures to evaluate the performance of our classifiers, i.e., precision, recall and F1-measure. Precision (P) is the proportion of actual positive class members returned by our method among all predicted positive class members returned by our method. Recall (R) is the proportion of predicted positive members among all actual positive class members in the data. F1 is the harmonic average of precision and recall which is defined as F1 = 2PR/(P+R).

4.2 Best Performers

Empirical results for intent classification are shown in tables 6, 7 & 8. Results for NER are shown in tables 9, 10, 11 & 12. Based on evaluation metrics, we can clearly see that the ensemble models outperform all other DNN models in both tasks, intent classification as well as NER by a good margin. In case of intent classification, the ensemble model (top 5) highlighted ○ in table 8 has the lowest count of false positives and false negatives on both the dev and test data sets. It also has the highest F1 score value = 99.91% beating the baseline rule-based system by a margin of ≈1.5%. In case of NER, the ensemble model (top 5) highlighted ○ in table 12 outperforms all other DNN models and beats the baseline model by a margin of ≈6%.

5. CONCLUSION AND FUTURE WORK

Our results show an ensemble model of stacking different DNNs of varying architectures outperforms individual performances of DNNs for the tasks of legal intent classification and entity recognition. RNNs, LSTMs, GRUs and even CNNs, all compress the necessary information of a source query into a fixed-length vector. This makes it difficult for the DNNs to cope with long queries, especially those that are longer than the queries in the training corpus. In future, we plan to use attention within queries. Attention is the idea of freeing a DNN architecture from the fixed-length internal representation. The DNN models we trained are at the word level, in future we plan to expand the size of the training data and try DNN models at the character level. Moreover, since the difference in performance between the DNN models were rather small, we plan to run tests of statistical significance and error analysis to capture performance by patterns. Lastly, we also plan to look into the impact on our models with respect to data and covariance shifts.

6. ACKNOWLEDGEMENTS

This research was supported by LexisNexis, Raleigh Technology Center, USA.

REFERENCES

[1] "http://www.legalexecutiveinstitute.com."

[2] L. Deng and J. C. Platt, "Ensemble deep learning for speech recognition," in *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pp. 1915–1919, 2014.

[3] X. Zhou, L. Xie, P. Zhang, and Y. Zhang, "An Ensemble of Deep Neural Networks for Object Tracking," in *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 843–847, Oct 2014.

[4] S. G. Soderland, "Building a Machine Learning based Text Understanding System," 05 2001.

[5] S. M. Beitzel, E. C. Jensen, O. Frieder, D. D. Lewis, A. Chowdhury, and A. Kolcz, "Improving Automatic Query Classification via Semi-supervised Learning," in *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pp. 8 pp.–, Nov 2005.

[6] D. Shen, R. Pan, J.-T. Sun, J. J. Pan, K. Wu, J. Yin, and Q. Yang, "Q2c@ust:Our Winning Solution to Query Classification in KDDCUP 2005," *SIGKDD Explorations*, vol. 7, pp. 100–110, 2005.

[7] S. Zhai, K. Chang, R. Zhang, and Z. M. Zhang, "DeepIntent: Learning Attentions for Online Advertising with Recurrent Neural Networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pp. 1295–1304, 2016.

[8] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.

[9] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *CoRR*, vol. abs/1412.3555, 2014.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pp. 1106–1114, 2012.

[11] X. Zhang and Y. LeCun, "Text Understanding from Scratch," *CoRR*, vol. abs/1502.01710, 2015.

[12] J. Hu, G. Wang, F. H. Lochovsky, J. Sun, and Z. Chen, "Understanding User's Query Intent with Wikipedia," in *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, pp. 471–480, 2009.

[13] H. B. Hashemi, A. Asiaee, and R. Kraft, "Query Intent Detection using Convolutional Neural Networks," in *WSDM QRUMS Workshop*, 2016.

[14] J. P. C. Chiu and E. Nichols, "Named Entity Recognition with Bidirectional LSTM-CNNs," *TACL*, vol. 4, pp. 357–370, 2016.

[15] N. Limsopatham and N. Collier, "Bidirectional LSTM for Named Entity Recognition in Twitter Messages," in *Proceedings of the 2nd Workshop on Noisy User-generated Text, NUT@COLING 2016, Osaka, Japan, December 11, 2016*, pp. 145–152, 2016.

[16] K. Sugathadasa, B. Ayesha, N. de Silva, A. S. Perera, V. Jayawardana, D. Lakmal, and M. Perera, "Legal Document Retrieval using Document Vector Embeddings and Deep Learning.," *CoRR*, vol. abs/1805.10685, 2018.

[17] R. Nanda, K. J. Adebayo, L. D. Caro, G. Boella, and L. Robaldo, "Legal Information Retrieval using Topic Clustering and Neural Networks," in *COLIEE 2017. 4th Competition on Legal Information Extraction and Entailment, held in conjunction with the 16th International Conference on Artificial Intelligence and Law (ICAIL 2017) in King's College London, UK.*, pp. 68–78, 2017.

[18] A. H. N. Tran, "Applying Deep Neural Network to Retrieve Relevant Civil Law Articles," in *Proceedings of the Student Research Workshop Associated with RANLP 2017*, (Varna), pp. 46–48, INCOMA Ltd., September 2017.

[19] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF Models for Sequence Tagging.," *CoRR*, vol. abs/1508.01991, 2015.

[20] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep Contextualized Word Representations.," *CoRR*, vol. abs/1802.05365, 2018.

[21] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality.," in *NIPS* (C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, eds.), pp. 3111–3119, 2013.