Regular Expression Guided Entity Mention Mining from Noisy Web Data

Shanshan Zhang, Lihong He, Slobodan Vucetic, Eduard C. Dragut
Computer and Information Sciences Department
Temple University, PA
{zhang.shanshan, lihong.he, vucetic, edragut}@temple.edu

Abstract

Many important entity types in web documents, such as dates, times, email addresses, and course numbers, follow or closely resemble patterns that can be described by Regular Expressions (REs). Due to a vast diversity of web documents and ways in which they are being generated, even seemingly straightforward tasks such as identifying mentions of date in a document become very challenging. It is reasonable to claim that it is impossible to create a RE that is capable of identifying such entities from web documents with perfect precision and recall. Rather than abandoning REs as a go-to approach for entity detection, this paper explores ways to combine the expressive power of REs, ability of deep learning to learn from large data, and human-in-the loop approach into a new integrated framework for entity identification from web data. The framework starts by creating or collecting the existing REs for a particular type of an entity. Those REs are then used over a large document corpus to collect weak labels for the entity mentions and a neural network is trained to predict those RE-generated weak labels. Finally, a human expert is asked to label a small set of documents and the neural network is fine tuned on those documents. The experimental evaluation on several entity identification problems shows that the proposed framework achieves impressive accuracy, while requiring very modest human effort.

1 Introduction

Named Entity Recognition (NER) is the task of automatically locating, extracting, and classifying contiguous pieces of strings, which represent entities of interest, in text. Classification (or typing) seeks to assign pre-defined categories (e.g., person, organization, location, expressions of time, monetary values, and emails) to each extracted piece of text. NER is a subtask of the broader problem of Information Extraction (IE) from text (Chang et al., 2006; Etzioni et al., 2005; Finkel and Manning, 2009; Nadeau and Sekine, 2007; Shen et al., 2015). Named entities usually refer to entity names that describe unique identifiers of people, locations, movies, events, and organizations. There is a large class of entities that are not "named," such as expressions of time, emails, and course identifiers. Their main characteristic is that they often follow an underlying syntactical pattern, which can be fully described or well approximated by Regular Expressions (REs).

Despite being the workhorse of many entity recognition tasks, REs have a number of drawbacks. The construction of highly accurate REs is difficult and requires specific technical skills. For a simple task such as recognizing emails, there are 361 REs proposed in RegExLib.com. Moreover, REs are brittle and difficult to maintain. These obstacles have motivated the work on automatic inference of REs (Banko et al., 2007; Li et al., 2008; Bartoli et al., 2018) where the objective it to develop approaches that are fast and deployable in real time. However, the existing approaches tend to require large number of examples to cover both the alphabet and the possible syntactic patterns. Moreover, they often produce overly complicated or long REs or combinations of REs (Bartoli et al., 2016). One of the most complete REs for emails has nearly 6,500 characters (Millner, 2008)!

Web documents are an important domain for data extraction. Importantly, the Web is not a place where data follows "underlying syntactical pattern" at scale. A datetime RE for New York Times news articles may not work for the articles at Le Figaro or Al Jazeera. Small typos throw off REs and produce nothing. This is a concrete Web example `data-timestamp="Thu Oct 05 2017 10:33:05 -0500">` that contains an unexpected space before "`-0500`". This small typo

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1991–2000
Brussels, Belgium, October 31 - November 4, 2018.

can easily deem a complex and painstakingly constructed datetime RE obsolete. At such scale, any attempt to fully understand an RE and debug it in case it fails is futile. New automated or semi-automated tools are needed to either supersede REs or to work in tandem with REs. In this work, we focus on the later.

In this paper, we target the problem of detecting presence of entity mentions that follow or closely resemble patterns that can be described by REs. Unlike much of the previous body of work on this topic, we do not focus on learning/inferring highly accuracte REs for entity identification (Prasse et al., 2012; Bui and Zeng-Treitler, 2014; Li et al., 2008; Banko et al., 2007; Brauer et al., 2011; Bartoli et al., 2016). We aim instead to show that deep learning can leverage imperfect REs and achieve very high accuracy while requiring only a modest human involvement.

Suppose the goal is to recognize *datetime* string expressions. We use some reasonable REs R for datetime to generate a weakly labeled training dataset from a large corpus of Web documents, e.g., news articles. We train a deep neural network on this data. Denote this model $\boldsymbol{M_{RE}}$. To our surprise, $\boldsymbol{M_{RE}}$ is already capable of recognizing the presence of datetime expressions beyond those recognized by R. Furthermore, with the addition of a very small number of training samples (between 20 - 50 instances) from a human labeler, we obtain a model $\boldsymbol{M_{RE+human}}$ that is superior to $\boldsymbol{M_{RE}}$ by a significant margin. In general, complex systems do not generalize easily with the addition of new data, because the amount of labeled data required to provide a good coverage grows exponentially with the complexity of the problem (Chiu and Nichols, 2015; Lample et al., 2016; Huang et al., 2015; Mahajan et al., 2018). We show that there is an opportunity for faster convergence to a generalized recognizer for this class of entities.

The main contributions in this paper are:

- We show how starting from REs R that recognizes a fraction of entities of a given type $\mathcal{E}$ (say, email) we can pretrain a deep neural network (NN) model which can be a richer recognizer of entities of type $\mathcal{E}$ than R.
- We show that we can fine tune the pretrained model to recognize an even larger set of entities of type $\mathcal{E}$ with the addition of a small number of labeled instances, as small as 20.

The paper is organized as follows. Section 2 gives an overview of the related work. Section 3 describes our method. Section 4 presents our methodology for parameter learning and experimental setup. Section 5 gives the experimental results. Finally, Section 6 concludes the paper.

2 Related Work

This section will mention several lines of research we deem the most related to our work.

The problem of inducing regular expressions has been an active area of work for more than two decades. One line of work focuses on improving the initial REs by identifying the true or false matches (Li et al., 2008; Murthy et al., 2012; Cetinkaya, 2007; Cochran et al., 2015). Another line of work attempts to directly induce REs from positive and negative sample strings (Fernau, 2009; Denis, 2001). The common approaches include generation of prefix and suffix automatons that represent overlapping syntactical features of the entities on character and token level (Brauer et al., 2011) and the automatic creation of REs based on genetic programming (Bartoli et al., 2012, 2014, 2016, 2018).

Constraining NN training to comply with known rules has also been an active research topic. Hu et al. (2016) proposes integration of constraints coming in the form of first order logic rules during training of NNs. Alashkar et al. (2017) trains an NN by minimizing a joint loss based on prediction of labels and adhering to the predefined rules. Locascio et al. (2016) proposed training LSTM NN to generate REs from sample pieces of text. Luo et al. (2018) incorporates knowledge of REs into training of NNs at three different levels: as the input features to NNs, as regularizations of the outputs of NN layers, or as a reward/penalty in the loss functions in NNs.

Unlike the aforementioned work, we do not attempt to learn explicit REs and do not force the outputs of NN layers match predetermined rules. Instead, we leverage REs as a means of generating a large quantity of weak labels from unlabeled data and using such data to pre-train an NN to recognize the provided REs. We fine tune such an NN on human-labeled data to exceed accuracy of the REs. A similar approach is effective in other domains. For example, Felbo et al. (2017) proposed pre-training an NN on millions of tweets labeled by emojis before fine tuning it for sentiment anal-

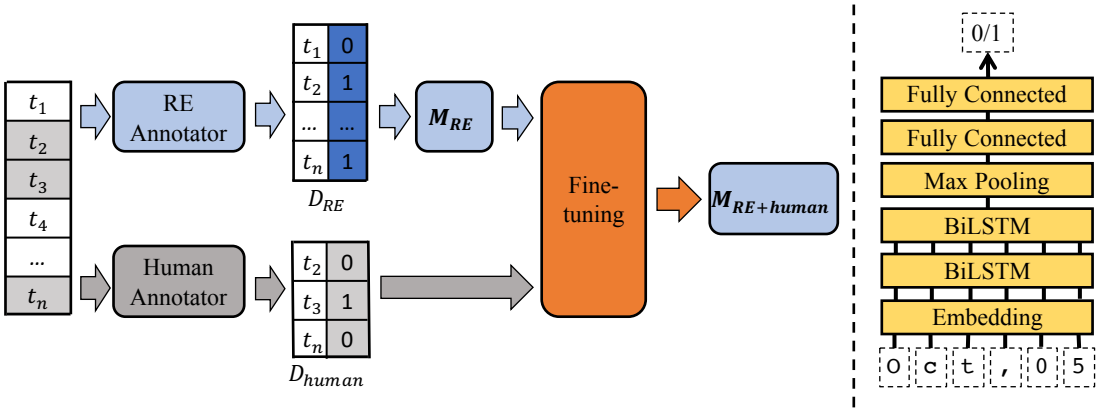

Figure 1: Overview of the solution (left) and deep learning architecture (right) used to train model $\boldsymbol{M_{RE}}$.

ysis. Mahajan et al. (2018) pre-trained an NN on billions of images labeled by hashtags before fine tuning it to various computer vision tasks.

3 Methodology

We describe the proposed framework here.

Problem Definition: Given a text string t and an entity type $\mathcal{E}$, the task is to predict whether t contains an entity mention of type $\mathcal{E}$. We treat this task as binary classification. To build the classifier, we assume that we are given a large corpus of unlabeled text strings $T = \{t_1, t_2, ..., t_n\}$. The challenge is to train the classifier with the minimal human effort. We allow a human expert to help in two ways: (1) construct a new RE or find an RE created by others, and (2) label an unlabeled string. For the purposes of this paper, we assume that one or more REs suitable for entity identification are already available and that human effort refers only to string labeling. The available REs might have an arbitrary precision and recall. We will analyze the impact of the RE quality on classification accuracy in the experimental section.

Solution Framework: An overview of the proposed framework is illustrated in Figure 1.

STEP 1. All of the unlabeled strings from T are fed into an RE annotator. If any of the provided REs t_i recognizes t_i, it is weakly labeled as $y_i = 1$, otherwise it is weakly labeled as $y_i = 0$.

STEP 2. An NN model $\boldsymbol{M_{RE}}$ is trained based on the weakly labeled data $D_{RE} = \{(t_i, y_i)|i = 1, 2, ..., n\}$. Given a large number of sample strings, it is expected that we can train an NN with high accuracy on D_{RE}.

STEP 3. A subset of m strings from T are sampled randomly and a human annotator labels each of the sampled strings. String t_i is labeled as $y_i = 1$ if the annotator recognizes an entity type $\mathcal{E}$ in t_i and as $y_i = 0$, if not. We denote the resulting strongly labeled data set as $D_{human} = \{(t_i, y_i)|i = 1, 2, ..., m\}$, where $m \ll n$.

STEP 4. The pre-trained NN $\boldsymbol{M_{RE}}$ is fine tuned with D_{human} data. We call the resulting NN $\boldsymbol{M_{RE+human}}$. For comparison, we also train a randomly initialized NN directly on D_{human}. We call this NN $\boldsymbol{M_{human}}$. The expectation is that the pre-trained NN captures very useful information about the entity type $\mathcal{E}$ and that fine tuning is more effective than training a new NN from scratch.

RE Annotator: Let us denote the set of REs available for a specific entity type $\mathcal{E}$ as R. R may be either created by human experts or generated automatically by tools like (Li et al., 2008; Bartoli et al., 2018), which require a human-labeled subset of T. Both approaches are human-intensive.

Deep Learning Architectures: We do not have a preference over any deep learning architecture to train $\boldsymbol{M_{RE}}$, as long as it can handle character-level inputs and produce binary outputs. We have no strict assumption about the strings in T. They may contain sentences from a formal news article, pieces of HTML code, or a mixture of formal texts and informal texts. For this reason, we treat t_i as a sequence of characters by default. NN architectures that meet our condition include but are not limited to: CNNs (Kim, 2014), BiLSTMs (Lai et al., 2015), and BiLSTM with self-attentions (Lin et al., 2017). For this paper we implemented a BiLSTM architecture. However, we also tested a CNN architecture, reaching similar conclusions. Our architecture contains an embedding layer to project each character into a vector, 2

Task	Data Size	Example	Labor
Date Time	761,002	data-timestamp="Thu **Oct 05 2017 10:33:05 -0500**"> <script src='/js/next-stories.20170925144113.js'>	2 days
Course Number	44,651	<body><h1>**CS 556** Interactive Software Systems /home.html>Dan R. Olsen Jr. Office: 3360 TM	1 week
Bill Date	49,002	(a) shall terminate on **30-09-2012.** ealth Service Act (42 U.S.C. 254c-15(c)(8)	-
Email Address	29,035	06:13:00 -0700 From:**phillip.allen@enron.com** To: Date:Mon, 23 Oct 2000 06:13:00 -0700 From: tom	1 week

Table 1: Dataset Summary.

BiLSTM layers to encode a sequence of character embeddings into a sequence of hidden vectors, and a max pooling layer followed by 2 fully-connected layers to project the hidden vectors into binary labels (Figure 1, on the right).

Fine-tuning: Once we have a model M_{RE} trained on weak labels, there are multiple ways to improve the weak model with human annotations to get $M_{RE+human}$. One common way is to freeze the parameters of all other layers of M_{RE} and fine-tune the last fully-connected layer (Donahue et al., 2014). Felbo et al. (2017) propose a *'chain-thaw'* strategy, which freezes all layers, then sequentially unfreezes and fine-tunes a single layer at a time. We exploit a less costly strategy as proposed in (Erhan et al., 2010), which uses the weights learned in M_{RE} to initialize $M_{RE+human}$, and start training $M_{RE+human}$ immediately with human annotations.

4 Experiment Design

We aim to answer three research questions in our experiments: **Q1**. Is it possible to train an accurate NN classifier with a limited number of human generated labels? **Q2**. What is the difference between REs and an NN pretrained on those REs? **Q3**. Does the quality of REs matter?

4.1 Data Sets

We use four datasets which are described in Table 1 in our experiments:

- Date Time: We download 6,000 news articles provided in the dataset One Week of Global News Feeds in Kaggle [1]. After chunking each news article, we get 761,002 strings.
- Course Number: The documents to produce chuncked strings are from The 4 Universities Data Set at CMU World Wide Knowledge Base (Web->KB) project[2]. This dataset has 44,651 strings.
- Bill Date: 600 US Congress bills from the THOMAS online database are used to identify the entity mentions of type datetime provided by (Bartoli et al., 2016). We generate 49,002 chuncked strings for this task. Each text instance contains the bill date (and time) and the location (index) of the datetime substring in the text.
- Email Address: The dataset is a collection of publicly available Enron email addresses from (Li et al., 2008; Brauer et al., 2011). It has 29,035 chuncked strings in total.

To answer the above 3 questions, we manually label 6,000 random strings from each data set. We create our training and testing data from this sample. The remaining strings are used to create D_{RE}. We list the human effort spent on labelling each data set in Table 1.

As an example of human annotations is the string `data-timestamp="Thu Oct 05 2017 10:33:05 -0500">Ex-dep`, which contains an entity of type datetime. The string `<scriptsrc='/js/prev-next-stories.20170925144113.js'defer>` is an example of a negative instance. We give examples of positive and negative instances in Table 1. The presence of an entity mention of a desired type is highlighted in **bold** in positive instances.

Our human annotated data sets are attached as supplementary materials.

4.2 Regular Expression Generation

The number and the source of REs used to train the deep model M_{RE} are listed in Table 2.

For the recognition of Date/Time entity

[1] https://www.kaggle.com/therohk/global-news-week

[2] http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data/

Task	$\|R\|$	Source	Labor
Date Time	25	Experts	5 hrs
Course Number	4	Li et al. (2008) Murthy et al. (2012)	-
Bill Date	5	Murthy et al. (2012)	-
Email Address	5	RE Library[3]	-

Table 2: Summary of REs.

mentions, we create 25 distinct REs in total (a member of our team who is familiar with REs generated the 25 REs, then the rest of the team checked these REs). For example, the RE `(20[0-9][0-9])(0[1-9]|1[0-2])(\d{2})([01][0-9]|2[0-3])([0-5][0-9])([0-5][0-9])` identifies datetime entities in the format `YYYYMMDDhhmmss`, while the RE `(20[0-9][0-9])-(0[1-9]|1[0-2])-(\d{2})T([01][0-9]|2[0-3]):([0-5][0-9]):([0-5][0-9])Z` identifies datetime entities in the format `YYYY-MM-DDThh:mm:ssZ`.

For the task of course number identification, we used four REs, one of which is borrowed from the results learned by ReLIE (Li et al., 2008), and the remaining three are from (Murthy et al., 2012). The regular expressions to extract the entity mentions of date are all from (Murthy et al., 2012). For email address, we use the top five REs from the RE Library[3] website.

4.3 Experimental Setup

Models in Comparison We compare 5 models on the 4 data sets: (1) ***Naive***, which always predicts 0, because 0 is the majority class on all 4 tasks. (2) ***RE***, which uses the set of REs R designed by experts or tools to weakly label the strings. Although we have multiple REs for each task, R can be any subset of the available REs. (3) $\boldsymbol{M_{RE}}$, which is the pretrained model of weak labels generated by R in model (2). (4) $\boldsymbol{M_{human}}$, which is the model trained with human annotations. (5) $\boldsymbol{M_{RE+human}}$, which is the fine tuned model $\boldsymbol{M_{RE}}$.

Evaluation Metrics For each data set, we divide the 6,000 strings with human annotations into 5 folds. We leave one fold as our test data. The training data is selected from the other 4 folds. We report four scores for each model: Accuracy (ACC), F1, Precision, and Recall. We report the average results over three random repetitions in Section 5.

Hyperparameters We use 100 dimensions in the embedding layer. We set the activation function in the first fully connected layers as `tanh`. The batch size is set to 300. We also add dropout layers after the embedding layer, the max pooling layer, and the first fully-connected layer to avoid overfitting, with drop out rate at 0.5. Our implementation is in PyTorch. We tune the learning rate (`lr`), the hidden units size (`nhidden`) in BiLSTM layers and the output size (`nfc`) of the first fully-connected layer by 5-fold cross validation using a random 6,000 sample from D_{RE}, for the sake of expediency. The ranges of selection are: `lr` $\in [0.0002, 0.0005, 0.001, 0.002, 0.004, 0.008, 0.015]$, `nhidden` $\in [50, 75, 100, 125, 150, 200]$ and `nfc` $\in [20, 50, 100, 200, 500]$. We use the *random search* algorithm proposed in (Bergstra and Bengio, 2012) that has been proved more effective than *grid search*. The hyperparameters used to train $\boldsymbol{M_{human}}$ and $\boldsymbol{M_{RE+human}}$ are identical to those used to train $\boldsymbol{M_{RE}}$.

We train 2 epochs for $\boldsymbol{M_{RE}}$ on weakly labeled data. $\boldsymbol{M_{human}}$ and $\boldsymbol{M_{RE+human}}$ are trained for 50 epochs on strongly labeled data.

5 Experimental Results

In this section, we evaluate the proposed framework with extensive experiments on the 4 entity recognition tasks. We use the empirical results to understand how the quality of initial REs impacts our conclusions.

5.1 Entity Mention Detection with Limited Human Annotations

We report the comparisons of the 5 models in Table 3 when we only have 20 human annotations. We use all the available REs in each task in this experiment. Comparing the last two rows of each task, $\boldsymbol{M_{RE+human}}$ always outperforms $\boldsymbol{M_{human}}$ by a large margin according to all 4 evaluation metrics on all 4 data sets. The F1, Precision and Recall scores are more than twice larger for all of tasks. The pretraining strategy is quite effective despite the very limited human annotation.

In addition, $\boldsymbol{M_{RE+human}}$ is much better than ***RE*** in the datetime and Course Number tasks. Its Recall scores increase by 19.1% and 14.4%, respectively, in the two tasks. This means the human

[3] http://www.regexlib.com/

Model Name	Date Time (%)				Course Number (%)			
	ACC	F1	Precision	Recall	ACC	F1	Precision	Recall
Naive	77.34	0.00	0.00	0.00	68.47	0.00	0.00	0.00
RE	90.65	76.77	88.00	68.33	71.64	59.91	54.01	67.28
M_{RE}	91.45	79.09	**89.01**	71.39	72.64	61.52	55.21	69.47
M_{human}	74.78	40.75	37.25	46.38	62.81	30.74	37.63	28.45
$M_{RE+human}$	**93.50**	**85.44**	87.03	**84.95**	**82.86**	**74.31**	**72.45**	**77.01**
	Bill Date (%)				Email Address (%)			
Naive	92.83	0.00	0.00	0.00	88.75	0.00	0.00	0.00
RE	**94.56**	38.23	**97.92**	24.01	**98.72**	**94.61**	**89.79**	**100.00**
M_{RE}	94.53	38.10	96.30	23.96	98.64	94.28	89.19	**100.00**
M_{human}	92.56	2.15	6.25	1.30	83.44	42.30	35.99	53.03
$M_{RE+human}$	94.36	**39.95**	87.88	**27.59**	97.75	90.95	83.60	**100.00**

Table 3: The comparisons in the presence of very few human annotations: $|D_{human}| = 20$.

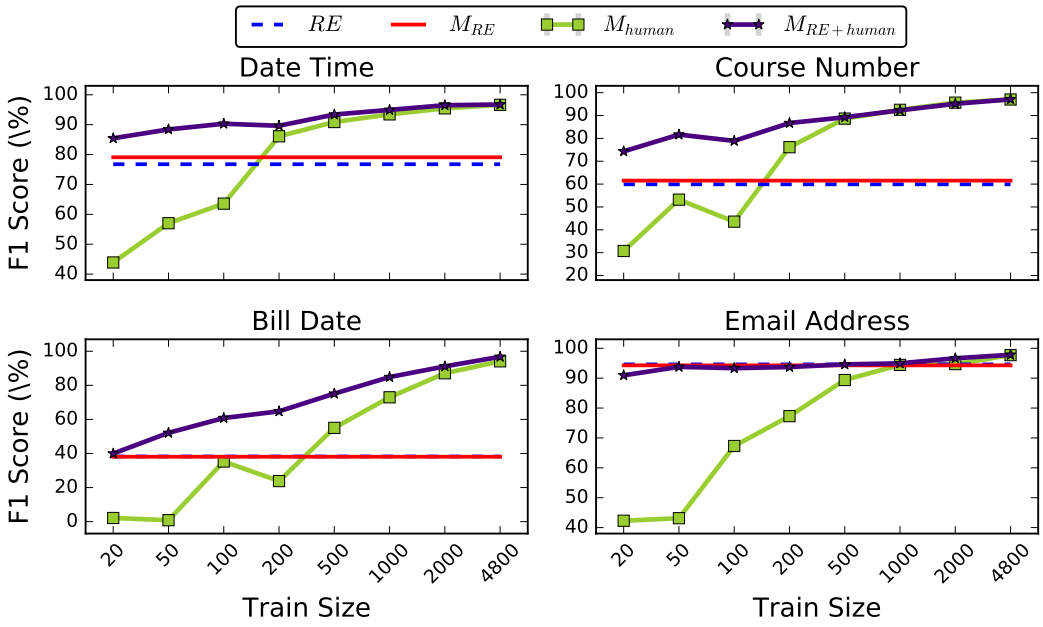

Figure 2: Trend of F1 when varying the number of human annotations.

annotations greatly increase the coverage of the initial REs for entity mentions. The Bill Date task is hard, since the initial REs already achieve Precision = 98% and entity mentions are really rare (ACC = 93% in ***Naive*** model). We still are able to improve the Recall by 15%, but at the expense of reduced Precision. This only gives a slight improvement in F1 (4.5%) and unchanged Accuracy. The Email Address task is even harder, with 90% Precision and 100% Recall from the initial REs. We fail to improve the accuracy with only 20 human annotations in this task.

To summarize, the answer to **Q1**, it is possible to train accurate NNs with limited amount of human generated labels and large amount of weak labels generated automatically.

5.2 Effect of the Number of Human Annotations

In Figure 2 we show how F1 varies with the size of the strongly labeled dataset, $|D_{human}| = [20, 50, 100, 200, 500, 1000, 2000, 4800]$. The observed trend is consistent over all 4 tasks: the larger the set of human annotations the better the performance of both M_{human} and $M_{RE+human}$. For the first 2 tasks, $M_{RE+human}$ surpasses all other competitors at 20; it takes 50 human annotations in Bill Date

	Date Time	
	Example	Labels
Group 1 19 / 11 / 8	"20170825" /><meta name="utime" content="**20170828042824**"/>	1 / 1 / 0
	{"origin":"mw1273","timestamp":"**20171005170835**","ttl":1900800}	1 / 1 / 0
	08/29/2017 12:13 pm ET	1 / 1 / 0
	2017/08/29/1/1700000000AEN20170829007751315F.html	0 / 0 / 1
	11-hristo-dimitrov-2017-10-05-06-57-492-c37d5a31-4ade	0 / 0 / 1
Group 2 48 / 36 / 12	0","dateModified": "**Thu, 05 Oct 2017 17:26:30 +0000**"	1 / 0 / 1
	<meta property="published",content="**2017-08-28T14:4316-0400**" />	1 / 0 / 1
	published **August 24, 2017 at 6:00 am**</time>	1 / 0 / 1
	http://www.businesswire.com/news/home/20170829005822/en/</p>	0 / 1 / 0
	resources/MWimages/MW-FV607-lava-2-MC-20171004095909.jpg">	0 / 1 / 0

Table 4: Examples of misclassified strings by $\boldsymbol{M_{RE}}$ and ***RE*** for Date Time task. The first columns shows number of misclassified strings, number of positives, and number of negaives. The last column represent human label and classifications by ***RE***, and $\boldsymbol{M_{RE}}$. Datetime is in bold for positive human annotation.

task, and $2,000$ in Email Address task.

5.3 Case Studies

From Table 3, we observe that $\boldsymbol{M_{RE}}$ achieves higher F1 scores than ***RE*** by about 2.3% on Date Time and 1.6% on Course Number tasks. This is a seemingly surprising outcome; NN trained on weak labels does better on human labeled test data than the REs used to generate the weak labels. To provide an insight, in Table 4 we illustrate example strings on which ***RE*** and $\boldsymbol{M_{RE}}$ disagree. *Group 1* consists of 19 strings where ***RE*** is correct and $\boldsymbol{M_{RE}}$ is not. *Group 2* consists of 48 strings where ***RE*** is incorrect and $\boldsymbol{M_{RE}}$ is correct. Looking at the examples in the first and last 2 rows of the table, we find strings that match RE with `YYYYMMDDhhmmss` format. This is an unusual form and it is expected that it might not always correspond to datetime entity. We hypothesize that the neural network encountered many negative strings in the weakly labeled data that have a similar form and learned that this form is not a reliable predictor of datetime.

Rows 6 - 8 in Table 4 illustrate the resilience of $\boldsymbol{M_{RE}}$ to small variations in the original REs. For example, despite *an extra space* in row 6, *a missing colon* in row 7, and *a mixture of spoken language* in row 8, $\boldsymbol{M_{RE}}$ is able to detect those entities, but REs are not. We give cases where $\boldsymbol{M_{RE}}$ makes mistakes in rows 3 - 5, showing that the NN is not able to learn the underlying REs with 100% accuracy.

To summarize, the answer to **Q2**, it appears that NNs are more noise resilient than REs.

5.4 Impact of the Initial REs

In this subsection, we investigate the impact of the choice of REs R on the accuracy of NN models. Since ***Naive*** and $\boldsymbol{M_{human}}$ models are not affected by R, we compare only three models in this subsection: ***RE***, $\boldsymbol{M_{RE}}$ and $\boldsymbol{M_{RE+human}}$. We select 4 out of the 25 REs in the datetime task for this study. The 4 REs are of different quality. We list the 4 REs in Figure 3. An example pattern that RE1 matches is `20180503101212`, for RE2 it is `2018-05-03 10:12:12`, for RE3 it is `2018-05-03T10:12:12Z` in UTC time zone and for RE4 it is `2018-05-03 10:12:12+00:00` or `2018-05-03 10:12:12-00:00`. We also consider the quality of NNs trained on weak labels from the whole set of 25 REs, denoted as All.

In Table 5 we compare the performance of ***RE*** and $\boldsymbol{M_{RE}}$ for the 5 different selections of REs for weak labeling. It can be observed that RE1 is the weakest individual RE in the group with F1 = 4.83, while RE4 is the strongest with F1$=41.86$. Using all 25 REs gives the highest accuracy of F1$=76.77$. We can see that $\boldsymbol{M_{RE}}$ closely follows the performance of ***RE*** and it is interesting to observe that $\boldsymbol{M_{RE}}$ becomes visibly superior only with good REs.

In Figure 3, we plot the 4 accuracy metrics for model $\boldsymbol{M_{RE+human}}$, which was pretrained on weakly labeled data generated by 5 different choices of REs, with varying initial RE sets and sizes of human annotations. We observe a significant influence of REs on accuracy. We can also observe that as the number of strong labels grows, the impact of RE choice decreases. When the number

Initial Set	*RE*				M_{RE}			
	ACC	F1	Precision	Recall	ACC	F1	Precision	Recall
RE1	75.88	4.83	24.89	2.68	75.71	4.20	23.68	2.32
RE2	79.97	20.72	100.00	11.58	80.00	20.92	100.00	11.70
RE3	79.78	19.34	100.00	10.75	79.83	19.75	100.00	10.99
RE4	83.40	41.86	100.00	26.71	83.73	43.77	100.00	28.19
All	90.65	76.77	88.00	68.33	91.45	79.09	89.01	71.39

Table 5: Comparison between ***RE*** and M_{RE} model for 5 different sets of REs.

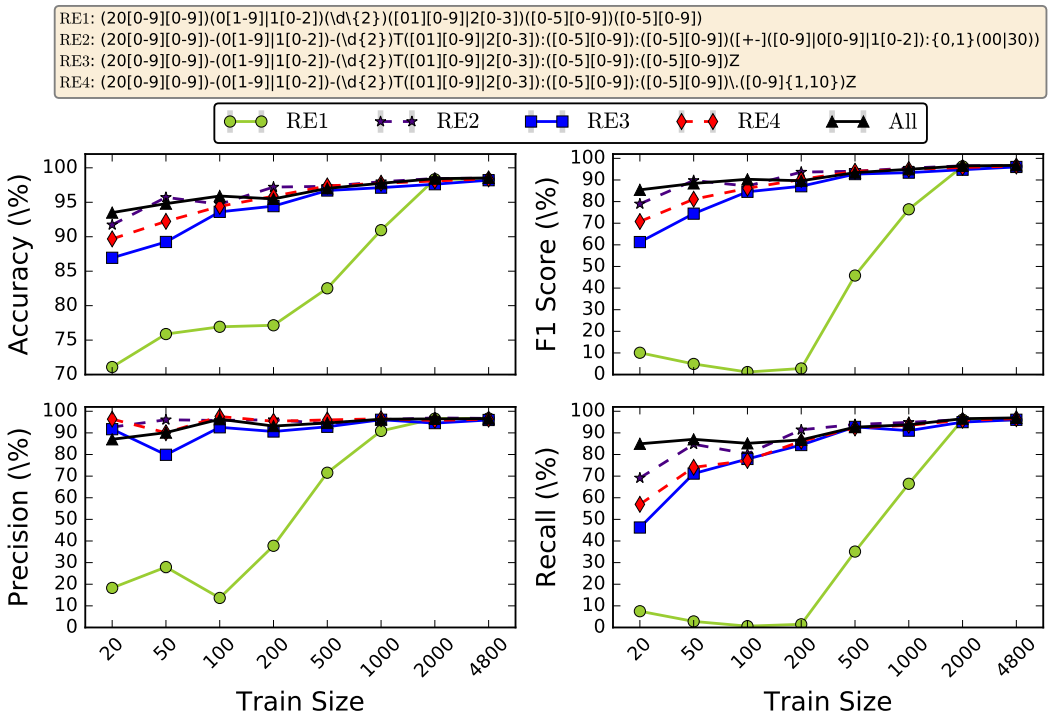

Figure 3: Performance of $M_{RE+human}$ for different initial sets of REs and sizes of D_{human}.

of strong labels exceeds $1,000$, the impact of the RE choice becomes negligible.

In summary, to answer **Q3**, there is a trade-off between creating more REs and creating more strong labels: (1) If designing a comprehensive RE takes a lot of time, a good strategy may be to take some time to construct one moderately good RE and spend more time on data labeling. (2) If the pattern is easy to describe by an RE, it may be a good strategy to spend time on creating a better set of REs and spend less time on labeling.

6 Conclusions

The main premise of this work is that it is practically impossible to create REs capable of identifying entities with perfect precision and recall at web scale. This paper explores ways to combine the expressive power of REs, ability of deep learning, and human-in-the loop into a novel integrated framework for entity recognition in web data. The framework starts by creating or collecting the existing REs for a particular type of an entity type (e.g., emails). Those REs are then used over a large document corpus to collect weak labels for the entity mentions and an NN is trained to predict those RE-generated weak labels. Finally, a human expert is asked to label a small set of documents and the neural network is fine tuned on those documents. The experimental evaluation on several entity identification problems shows that the proposed framework achieves impressive accuracy, while requiring very modest human effort.

Web sources often change in ways that prevent the induced REs from extracting data correctly. At the web scale, we require automated tools to maintain them. One direction of future work is to use our framework to diagnose when a RE is broken over a text stream.

Acknowledgments

This work was supported in part by the following grants: U.S. NSF BigData 1546480 and U.S. NIH R21CA202130.

References

Taleb Alashkar, Songyao Jiang, Shuyang Wang, and Yun Fu. 2017. Examples-rules guided deep neural network for makeup recommendation. In *AAAI*, pages 941–947.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*, volume 7, pages 2670–2676.

Alberto Bartoli, Giorgio Davanzo, Andrea De Lorenzo, Marco Mauri, Eric Medvet, and Enrico Sorio. 2012. Automatic generation of regular expressions from examples with genetic programming. In *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*, pages 1477–1478.

Alberto Bartoli, Giorgio Davanzo, Andrea De Lorenzo, Eric Medvet, and Enrico Sorio. 2014. Automatic synthesis of regular expressions from examples. *Computer*, 47(12):72–80.

Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao. 2016. Inference of regular expressions for text extraction from examples. *IEEE Transactions on Knowledge and Data Engineering*, 28(5):1217–1230.

Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao. 2018. Active learning of regular expressions for entity extraction. *IEEE transactions on cybernetics*, 48(3):1067–1080.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305.

Falk Brauer, Robert Rieger, Adrian Mocan, and Wojciech M Barczynski. 2011. Enabling information extraction by inference of regular expressions from sample entities. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1285–1294. ACM.

Duy Duc An Bui and Qing Zeng-Treitler. 2014. Learning regular expressions for clinical text classification. *Journal of the American Medical Informatics Association*, 21(5):850–857.

Ahmet Cetinkaya. 2007. Regular expression generation through grammatical evolution. In *Proceedings of the 9th annual conference companion on Genetic and evolutionary computation*, pages 2643–2646. ACM.

Chia-Hui Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. 2006. A survey of web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1411–1428.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.

Robert A Cochran, Loris D'Antoni, Benjamin Livshits, David Molnar, and Margus Veanes. 2015. Program boosting: Program synthesis via crowd-sourcing. In *ACM SIGPLAN Notices*, volume 50, pages 677–688. ACM.

François Denis. 2001. Learning regular languages from simple positive examples. *Machine Learning*, 44(1-2):37–66.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1):91–134.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. *arXiv preprint arXiv:1708.00524*.

Henning Fernau. 2009. Algorithms for learning regular expressions from positive data. *Information and Computation*, 207(4):521–541.

Jenny Rose Finkel and Christopher D Manning. 2009. Nested named entity recognition. In *EMNLP*, pages 141–150. Association for Computational Linguistics.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Yunyao Li, Rajasekar Krishnamurthy, Sriram Raghavan, Shivakumar Vaithyanathan, and HV Jagadish. 2008. Regular expression learning for information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 21–30. Association for Computational Linguistics.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Nicholas Locascio, Karthik Narasimhan, Eduardo DeLeon, Nate Kushman, and Regina Barzilay. 2016. Neural generation of regular expressions from natural language with minimal domain knowledge. *arXiv preprint arXiv:1608.03000*.

Bingfeng Luo, Yansong Feng, Zheng Wang, Songfang Huang, Rui Yan, and Dongyan Zhao. 2018. Marrying up regular expressions with neural networks: A case study for spoken language understanding. *arXiv preprint arXiv:1805.05588*.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. 2018. Exploring the limits of weakly supervised pretraining. *arXiv preprint arXiv:1805.00932*.

Rachel Millner. 2008. Four regular expressions to check email addresses.

Karin Murthy, P Deepak, and Prasad M Deshpande. 2012. Improving recall of regular expressions for information extraction. In *International Conference on Web Information Systems Engineering*, pages 455–467. Springer.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Paul Prasse, Christoph Sawade, Niels Landwehr, and Tobias Scheffer. 2012. Learning to identify regular expressions that describe email campaigns. In *Int. Conf. Mach. Learn.*, pages 441–448.

W. Shen, J. Wang, and J. Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460.