A Survey on Named Entity Recognition Methods Across Domain-specific Applications

Aaron Reich **Tanvi Bhagwat** **Sanmeshkumar Udhayakumar**

Department of Computer Science
Georgia Institute of Technology
Atlanta, GA 30308

1 Introduction

For our survey we investigated the research area of Named Entity Recognition. Named Entity Recognition (NER) is recognizing and extracting mentions of named entities in text. It can be used to label data in customer support tickets, detect entities mentioned in customer feedback, and easily extract important information, like contact information, location, dates, among other things. It could be used to create targeted news content by extracting the major entities present in an article. NER is important for the related task of Entity Linking as well as for downstream tasks such as question answering, knowledge graph population, and information retrieval.

There are existing surveys in the field of NER that mainly focus on comparing the performance of different deep learning architectures. However our survey will focus on challenging research directions in the field, such as semi-supervised deep learning approaches to limited labeled data in NER, challenges in domain-specific NER, and NER on noisy text in Tweets. To our knowledge a previous survey has not focused attention on these research areas and our survey will contribute a comprehensive evaluation of current approaches to these problems.

2 Method

NER's traditional approaches include rule-based approaches, unsupervised learning approaches, and feature-based supervised learning approaches. The newer methods are deep learning based approaches. We focused on these. [1]

Deep learning techniques benefit from the ability to apply non-linear transformations from the input to the output. This allows the model to learn complex features from the text that the traditional methods such as HMM's and CRF's are not able to learn. The deep learning based NER architecture consists of distributed representations for input, the context encoder, and the tag decoder. The distributed representations for input include design choices such as character-level embeddings, word-level embeddings, and feature engineered ones like gazetteer [2] and POS tags. The context encoder consists of the design choice of the type of neural network model to be used for the capturing of the context dependencies. The tag decoder is then used to predict the associated tags for the sequence depending on what tag scheme is used, for example B(begin) I(Inside) O(outside). [1]

Even within these representations, there are variations of techniques used. Within Stage 1 of distributed representations for input, the representations could be char-level, word-level, or a hybrid version. Within the char-level representations, some methods are LSTM, CNN, GRU, etc. Within word-level representations, some methods are GloVE, and Word2vec, among others. For stage 2 context encoder architectures, one could use CNN, GRU, LSTM, etc. And for stage 3 tag decoder, some possible methods are CRF, GRU, and LSTM. [1]

These varying deep learning NER architectures will be utilized in the different research areas of this survey.

2.1 Semi-Supervised Deep Learning Approaches to Limited Labeled Data in NER

2.1.1 Overview of Semi-Supervised Approaches to NER

NER is usually trained on one part of the dataset, and then not generalized for the test dataset. This leads to poor NER performance. There are a few reasons for this. To begin with, the corpus is not sufficiently labelled. Labelling a corpus for training is time consuming as it involves both identifying the NER string, and classifying it. [1] One approach is the use of data augmentation techniques. Data augmentation is formalized by the Vincinal Risk Minimization principle [3]. A neighborhood can be created around the data points in the training data[3]. Virtual examples can then be drawn from this vicinity distribution increasing the support of the training distribution[3]. Data augmentation allows for an improvement of generalization. Mixup involves the linear interpolations of the feature vectors in the training data which leads to linear interpolations of the respective labels[3]. We investigate the Local Additivity based Data Augmentation (LADA) approach for NER based on the mixup technique. Another semi-supervised approach involves the use of multitask variational methods[4]. One example is the use of a latent-variable generative model to model the conditional distribution of a word given its context and a discriminative labeler to input discriminative information into the latent space[4]. The last semi-supervised approach we investigate is the modeling of noise in the Mean Teacher model by adding Gaussian noise to input embeddings or by replacing words with their synonyms [5].

2.1.2 Local Additivity based Data Augmentation

In [6] they propose a Local Additivity based Data Augmentation (LADA) approach for NER based on the mixup technique. LADA has two variations. Intra-LADA interpolates each token's representation with other tokens in that chosen sentence for the purpose of increasing robustness towards different word orderings. Inter-LADA samples different sentences for its interpolation. They then made an extended semi-supervised LADA by the use of a novel consistency loss for the unlabeled data. For LADA, a sampled pair of sentences composed of word embeddings are inputted into a m-th layer encoder. The hidden representations of each of the sentences are then linearly interpolated with each other by a ratio sampled from the Beta distribution. After the interpolation, the hidden representations are then given as input to the upper layers of the L-layer encoder. The corresponding labels are also mixed at the same ratio. The output of the Lth layer encoder is the hidden representations of the virtual sample which is then passed into a classifier [6].

For Intra-LADA, sampling of the reordered sentences is from the distribution of 1 over the number of possible permutations of the original sentence. For Inter-LADA, a weighted combination of two k-nearest-neighbors and random sampling is used. When the sampled sentence is sampled from the entire training corpus, it causes large amounts of noise but acts as regularization on the model. However, when the sentence is sampled using k Nearest Neighbor, the sampled sentence shares a similar but different context as the original sentence thus providing a good signal to noise ratio. For the semi-supervised version of LADA, training samples are back translated to create paraphrases of the original samples. The entities remain the same in the paraphrased sentences as in the original sentence and no new entities are added. Their proposed novel consistency loss is then used for incorporating a sum of the supervised loss and the unsupervised loss with a hyper-parameter for controlling the tradeoff between each of the loss terms. Their experimental set up involves the use of pre-trained Flair embeddings and a multi-layer Bi-LSTM as the encoder for the first LADA model. The second LADA model uses BERT as the encoder and a linear layer for the classifier [6].

2.1.3 Variational Sequential Labelers

So while the Local Additivity based Data Augmentation approach uses a L-layer encoder and different mixup techniques, multitask variational methods instead utilize a variational autoencoder architecture. In [4], they employ multitask variational methods with the use of Variational Sequential Labelers (VSL) for the NER semi-supervised problem. In their work they combine a latent-variable generative model with a discriminatively-trained labeler. This modeling approach to the semi-supervised NER problem differs greatly from the sampling and mixup techniques of [6]. The learning associated with this model involves the maximization of the conditional probability of a word given its context

and the minimization of the classification loss given the latent variable space [4]. The models they explore are semi-supervised because of their ability to combine unlabeled and labeled data which results from the fact that they are generative latent variable models [4].

The Variational Sequential Labelers objective is the locating of the information that is valuable for predicting a word from its context. When only taking into consideration labeled data, this variational loss adds on regularization to the supervised sequence prediction model and is the unsupervised objective for the data that is not labeled. The generative model they use for the VSL consists of a feed-forward neural network with two hidden layers and ReLU as the activation function. For the inference model, they first use a Bi-directional gated recurrent unit in order to produce a hidden vector at each position. The input to it being the concatenation of the word embeddings and their associated hidden states from a character-level BiGRU. The hidden vectors are then used as input to a single layer feed-forward network [4].

They explore different latent variable configurations. One specific latent variable configuration they explore involves latent variables' hierarchical relationships. This model is called the VSL-GG-Hier model. It consists of two Gaussian latent variables and the model utilizes the fact that latent information corresponding to the specific word is different depending on the class information [4].

2.1.4 Mean Teacher with Noise Modeling

While the multitask variational approach uses a variational autoencoder architecture, the mean teacher approach instead uses Bi-LSTM's and multilayer perceptron networks. The use of the modeling of noise in the mean teacher approach proposed in [5] is similar to the use of noise in the Inter-LADA technique of [6]. However the noise technique in [6] involved the use of random sampling of a sentence from the entire training corpus. This is different from the noise techniques employed in the mean teacher approach which are direct changes to the data or the dropout of neurons from the student and teacher model networks. The two models used in the Mean-Teacher with Noise approach are the student and teacher. They have the same structure but the student's parameters are updated using back-propagation while the teacher's are updated as a weighted average of the student's parameters across the epochs. The cost function consists of the supervision loss and the consistency loss. The consistency loss is the L2 norm of the difference between the representations of the teacher and student models. The purpose of the teacher's parameters being the weighted average of the student's parameters and the inclusion of consistency loss is for the reduction of confirmation bias in the teacher model. This is due to the fact that the teacher model's predictions are used a pseudo-labels for the student model in training [5].

In both the student and teacher models, the tokens are represented as pre-trained word embeddings and a Bi-LSTM is ran on the sentence's unlabeled tokens as well as on the entity mention that consists of a label in the sentence. The entity mention and context final Bi-LSTM state representations are concatenated and used as input to a multi-layer perceptron with a single hidden layer for the producing of the output layer. Different forms of noise are employed in order to produce robust models and to deal with the constraint of limited labeled data available. Linguistic inspired noise includes the use of dropping words or replacing words with their synonyms based on inverse document frequency. Statistical noise is employed by applying Gaussian perturbations to pre-trained word embeddings. The final type of noise used is network noise by the use of dropout in the inner layers of the student and teacher model networks [5].

2.1.5 Comparison and Evaluation of Semi-Supervised Approaches to NER

Model	5%	10%	30%
VSL-GG-Hier	83.38	84.71	85.52
Mean-Teacher + Noise	82.60	83.47	84.88
BERT + Semi-Intra-LADA	87.15	88.70	89.69
BERT + Semi-Inter-LADA	86.51	88.53	90.00
BERT + Semi-Intra&Inter-LADA	86.33	88.78	90.25

Table 1: The F1 scores on the CoNLL 2003 with 5%, 10%, and 30% of the original training set. Each dataset consisted of 10,000 unlabeled data points which were randomly sampled from the original training set. The resulting F1 scores were averaged over 5 runs. This data is from [6]

As can be seen in Table 1, the variations of BERT + Semi-LADA outperform the other two models on the semi-supervised CoNLL challenge. VSL-GG-Hier slightly outperforms Mean-Teacher + Noise. Semi-LADA's back translation technique combined with the Inter-LADA and Intra-LADA approaches allows for a combination of a good signal to noise ratio while at the same time providing regularization to the model. The consistency loss used then offers an effective tradeoff between the supervised loss and unsupervised loss [6].

The data augmentation techniques of the Mean-Teacher + Noise model may be too rudimentary to be able to compare in performance to the other models at the semi-supervised NER tasK. The use of statistical noise such as applying Gaussian perbutations to word embeddings and linguistic noise such as dropping or replacing words does not compare in effectiveness to the back translations and linear interpolations of training samples employed by Semi-LADA. VSL-GG-Hier's VSL variational objective of locating the information that is valuable for predicting a word from its context serves as an effective form of regularization for the supervised learning aspect and as an effective unsupervised objective for the unlabeled data. The explicit hierarchical modeling of the Gaussian latent variables also allows it to outperform the Mean-Teacher + Noise model. However, these strengths may not be enough to achieve the performance of Semi-LADA which may be due to VSL-GG-Hier's lack of data augmentation techniques.

2.2 NER Performance For Specific Domains

In this section we will focus on where NER is today in terms of its application in specific domains, some of the challenges faced in this process, as well as solutions that have been explored in various papers. NER has been applied in various domains, such as Electronic Health Records (EHR) [7], biology [8], and the game domain [9]. The applications of NER in specific domains is infinite, and exploring the research done in this space will truly help get closer to implementation of NER in enterprise-grade systems. There are several problems that are hindering this currently however.

2.2.1 Availability of Domain-specific NER corpora

One problem is that often times, there are little or no availability of both annotated, and unannotated NER corpora readily available that is from the specific application domain. One reason for this is the paucity of domain-specific documents in languages other than English since most corpora for NER are in English. Another reason for the lack of availability of Domain-specific NER corpora is that NER tasks for specific domains often have unique entity tags that aren't in generalized NER corpus. For a domain, NER tags could be as complicated as medical conditions such as "ischaemic stroke" and "glioma tumour", which have little to no probability of appearing in generalized NER corpus. Figure 1 shows different tag-sets belonging in different domains, organized from left to right in increasing tag specialization.

One solution to deal with the lack of availability of domain-specific NER corpora is to collect domain-specific documents from the web or from other domain-specific resources to use as a large corpus of unannotated data, do pretraining of word embeddings on the unnannotated corpus, label a small subset of the corpus manually, and then train the DNN (Deep Neural Network) on that. The advantage of this methodology is that the data used for unannotated domain-specific corpus is often widely available. Refer to Table 2 to see the different sets of unlabelled and labelled data to do different domain-specific NER tasks, in order to get an idea of the variety of data that can be used for the unsupervised corpus. Thus, you can do pretraining of word embeddings on a large unannotated dataset, and then limit the time and domain-expert resources required in labelling the data for the full DNN model training and evaluation by only needing to label a small corpus. Several publications have shown that this methodology has given rise to higher domain-specific NER accuracies [7] [8], with F1 scores increasing by up to 15% in [7] when using word embeddings pretrained on domain-specfiic corpus instead of general corpus.

Another approach that you can do in addition to pretraining the word embeddings is to do transfer learning, as in the case of [10]. More specifically, you can initialize the DNN weights with that of another model that may share at least some of the tag-sets that you have for your domain. The advantage of this approach is that transfer learning has been shown to increase the the accuracy dramatically when working with little annotated data, and when combined with pretraining of word embeddings method, it's shown in [10], that the F1 score of a method using pretraining and transfer learning with 50% of a dataset used for training outperforms a model with no transfer learning that

Publication Reference	Task	Unlabelled Corpus Sources for pretraining of word-embeddings	Labelled Corpus Sources
[7]	EHR	• Wikipedia biomedical category pages, • Package leaflets of drugs, • Biomedical scientific articles in Italian, • Italian Medical Dictionary	Manually Labelled Italian EHR
[10]	EHR	• Sample medical notes • drug public assessment reports • FAQ sections	Manual Annotation of subset of unlabelled data
[11]	EHR	anonymised radiology reports from brain MRI • and CT scans	Manual Annotation of subset of unlabelled data
[8]	Bacteria	• Publication abstracts from PubMed	Manual Annotation of subset of unlabelled data
[9]	Shogi Commentary	• Subset of labelled data	Shogi commentary corpus with associated game state information

Table 2: Unlabelled and labelled corpus sources for 5 different domain-specific NER tasks. Unlabelled corpora are used for pretraining of word-embeddings, and labelled corpora are used for training and evaluation of full DNN model

uses 100% of the training data. The disadvantage of this method as mentioned before is that you might not have any other model that shares any tags with your domain dataset, and the tag for a specific word in the generalized corpus context, might be different than for that word in a specific domain, as shown with the Baltimore example reviewed previously.

Another approach to help fix the issue of there often times being little to no annotated data available is to supplement the text corpus used for NER with other real-world data that corresponds to the text corpus. For example, in [9], in the application of NER for Shogi game commentary, the Shogi game commentary transcript corpus is supplemented with game state information of the corresponding Shogi game. This game state information essentially records the configuration of the different Shogi pieces on the board for each player. Other examples of using alternative real-world data to help with an NER task on textual data is to use images or video that correspond to text that describes it. As per how the real-world data is incorporated in a DNN model, [9] trains a stacked autoencoder on just the game states to create embeddings for the game states to feed into the DNN, along with the regular word and char embeddings. The advantages of this method is it provides a solution to increase accuracy when there is little annotated data available. Also it can be done in addition to the other previously mentioned methods of pretraining and transfer learning without confilict. The disadvantage of this method, is that it hasn't been shown to provide a large increase in F1% score, compared to other methods like pretraining. In [9], it only increased the NER F1 score by 0.64%, from 84.10% to 84.74%.

2.2.2 Named Entity Name Complexity

Often times, NER can be hard in domain-specific applications because of the complexity of the names used that we want to tag. For example in [8], we would have to tag phrases as complicated as "Actinobacillus actinomycetemcomitans, Porphyromonas gingivalis, and Peptostreptococcus micros", where we would have to recognize "Actinobacillus actinomycetemcomitans", "Porphyromonas gingivalis", and "Peptostreptococcus micros" as separate bacteria entities. Thus in order to solve this problem, for bacteria NER, [8] proposes combining domain features into the deep learning model, specifically using encodings generated from POS (part of speech) tagger, as well as encodings generated from text matching a dictionary of known bacterial names and abbreviations drawn from the UMLS unified medical database. To generate the dictionary embeddings for a word in the corpus, and thus in order to find matching text sections in the dictionary, the BDMM (bidirectional maximum matching algorithm) is employed for word segmentation [12]. Typically in the embedding layer, only character and word embeddings are used, but with the addition of domain-specific features of pos embeddings and dictionary embeddings, it was shown in [8] that the F1 score went up from 84.654% to 89.700%, which is a significant increase. The downside of this method is that it requires you to have a dictionary, or at least take the time to create one, which might not be practical.

Publication Reference # and f1 score	[7] F1 score %	[10] F1 score %	[11] F1 score %	[8] F1 score %	[9] F1 score %
Application	EHR	EHR	EHR	bacteria	Shogi
General Corpora Pretraining	85.45%	70.40%			
Domain-specific pretraining	96.61%		80%	89.15%	84.10%
Pretraining + pos				89.41%	
Pretraining + pos + dict				89.70%	
Transfer learning		74.90%			
Transfer learning with pretraining		78.50%	89%		
Pretraining + Real-world data					84.74%
Main Technique analyzed	Domain-specific pretraining	Pretraining + transfer learning	Pretraining + transfer learning	pos+ dict	real-world data
F1 score increase from base case	11.16%	8.10%	9%	0.55%	0.64%

Table 3: Comparison of the different techniques to improve domain-specific NER F1 score across different publication experiments.

2.2.3 Deciding Methodology to Achieve NER for Domain-Specific Applications

When a NER task is taken on for a specific domain, it is often debated which methodology should be used in order to achieve the goal: rule-based method, a deep learning method, or a deep learning with transfer learning method. [11] explores this in the EHR (Electronic Health Record) domain and makes some key findings that can be useful in deciding which approach to pursue. In [11], the F1 scores of each method were: rule based method = 93%, deep learning method = 80%, deep learning with transfer learning = 89%. It was found that the rule-based method performed with the best F1 score overall. However, the downsides were that it performed significantly bad on a few tags, with multiple tag F1 scores being below 30% unlike the other two DNN methods. This makes sense, because rules are usually designed in such a way that they either fully capture the pattern to do NER for certain tags, or they largely miss the pattern for NER of certain tags. Also, it takes a lot of time and domain expert resources in order to manually create the rules.

Deep learning method with pretraining of word embeddings, but with no transfer learning, performed significantly worse than rule-based methods, but had a more consistent F1 score performance than the rule-based method. Deep learning methods with transfer learning only had 4% difference in F1 score, and thus performs very similarly to rule-based method. Another advantage of Deep learning with transfer learning method over rule-based methods was it had a more consistent F1 score across all classes, with the 2 lowest F1 scores for a particular tag being 63% and 78%. The downsides of deep learning with transfer learning are that it requires a annotated dataset, and it has less interpretability than rule-based methods, and thus can make it harder to debug than rule-based methods.

2.2.4 Summary of different Domain-specific NER techniques

Below is a table summary of the different techniques used to improve domain-specific NER performance, and the pros and cons of each method has been covered in their respective subsections. As we can see from Table 3, domain-specific pretraining and transfer learning are the most impactful as well as prevalent methods used. Currently, using part of speech tagger and dictionary, as well as augmenting the dataset with real-world data hasn't increased the F1 score significantly, but we hope further research in this area will further this improvement.

2.3 NER on Noisy Text in Tweets

Tweets are harder to classify due to their shortness and the lack of contextual information. Because of their noisy nature, they may also contain Named Entities not a part of any gazetteer. Often due to their up-to-date nature, they may even reference entities that haven't been mentioned on Wikipedia, making the task of NER all the more harder [13].

TwiNER presents an unsupervised NER system for targeted tweet streams. TwiNER does not categorize the type of the named entity e.g. person or location, but rather identifies the segments.

Since tweets can't make use of the traditional linguistic features such as capitalization or punctuation, this approach uses another property instead. It was noticed that the tweets still contained the correct collocation of a named entity in a tweet. Using this, the tweets are segmented into a collection of consecutive phrases. Instead of counting the probability of a word, the probability of a segment is calculated. An example tweet, "my shoes are gg to compete in the youth olympic games sailing competition. It just needs a mast and a rudder", can be separated into the following segments after removing the stop words, "(shoes) | (gg) | (compete) | (youth olympic games) | (sailing competition) | (needs) | (mast rudder)". Each segment within this tweet is considered to be a a named entity. To find the true named entity, confidence scores are calculated. To split the tweet into segments, dynamic programming can be used over the sequence of data. To check if a segment should be split further, its collocation is checked. In TwiNER, the local and global contexts are summed up, to check for the word collocation.

TwiNER simulated tweets relating to two different events: tweets based on the top-1000 Twitter users in Singapore (SIN), and tweets based around the Singapore General Election (SGE). TwiNER achieved a precision of 92.9, recall of 66.0 and a F1 score of 77.2 on the SGE dataset. It achieved a precision of 41.9, recall of 32.9 and a F1 score of 41.9 on the SIN dataset [13].

While TwiNER only identifies segments for possible entities within a tweet, the second paper by Limsopatham et. al. uses a BiLSTM to identify 10 NER categories: company, facility, geolocation, music artist, movie, person, product, sports team, tv show and other entities. This approach is made of three main components. The first component, the orthographic sentence generator, creates an orthographic pattern of words for each input sentence. For e.g. "14th MENA FOREX EXPO announced!!" is now "nnCC CCCC CCCCC CCCC ccccccccpp" where the upper-case, lower-case, and the numbers and the punctuations are replaced with C, c, and p. The next component, word representation as input vectors, is created from the word-level and character-level representations for each word and its orthographic representation. The third component, bidirectional LSTM, is used to learn the contextual information given within the sentences.

This experiment was conducted on the shared task at the 2016 WNUT workshop. A precision of 73.49, recall of 59.72 and a F1 score of 65.89 on the segmentation only task, but lower scores for precision 60.77, recall of 46.07 and a F1 score of 52.41 for the segmentation and categorisation task [14].

De Neve et. al. published a paper that also identified the same categories as Limsopatham et. al., but instead of a BiLSTM they used a feed forward neural network. This approach also has three steps. The words are converted into word representations. To convert the words into word embeddings, Word2Vec is used. The algorithm iterates over the dataset, updating the embeddings using a Skip-gram architecture and negative sampling. For out-of-dictionary words, an embedding of zeros is used. Next, these are fed to a Feed-Forward Neural Network (FFNN). Each entity will have a Begin and Inside tag. The FFNN will asign tags to each word out of the possible 21 different tags (Begin, Inside, Outside NE). Once, all the tags have been assigned, postprocessing is done, some of the tags are changed depending on the rules. If the NE does not start with a Begin tag, then we select the word before this word as beginning the tag. If the individual words within NE have different categories, the most frequently occurring category is selected.

This experiment was evaluated as a part of the shared task at ACL 2015. Post-processing the F1 score was 49.09% against a baseline score of 34.29% for categorizing named entities. Post-processing the F1 score was 60.80% against a baseline score of 52.63% for detecting named entities [15].

If the previous two papers categorized a lot of entities, the fourth paper by Cherry et. al. only categorizes person, location and organization. This paper used a discriminative, semi-Markov tagger. For this word-based model, the *Begin, Inside, Last, Outside* and *Unique* tags are used. This paper utilized the Word2Vec skipgram architecture, the Brown clustering algorithm, and a mixture of the two methods. The model is trained with a structured version of the Passive Aggressive (PA) algorithm. PA separates correct sequences from incorrect ones by a margin of 1. The model is updated one training sentence at a time. The feature function is based on the semi-Markov dynamic program. Examples from the in-domain pool are assigned a higher weight, than the ones from the out-of-domain pool.

System	Fin10Dev	Rit11	Fro14	Avg
CoNLL	27.3	27.1	29.5	28.0
+ Brown	38.4	39.4	42.5	40.1
+ Vector	40.8	40.4	42.9	41.4
+ Reps	42.4	42.2	46.2	43.6
Fin10	36.7	29.0	30.4	32.0
+ Brown	59.9	53.9	56.3	56.7
+ Vector	61.5	56.4	58.4	58.8
+ Reps	64.0	58.5	60.2	60.9
CoNLL+Fin10	44.7	39.9	44.2	42.9
+ Brown	54.9	52.9	58.5	55.4
+ Vector	58.9	55.2	59.9	58.0
+ Reps	58.9	56.4	61.8	59.0
+ Weights	64.4	59.6	63.3	62.4

Table 4: Evaluation of the experiment under 3 data scenarios

Table 4 shows the different values of F1 when the model is training on three different systems vs their performance on those systems, across the performances when considering different kinds of word representations, where Vector is Word2Vec, Brown is for Brown clustering, and Reps is for the hybrid combination of both [16].

The fifth paper we considered, by Tran et. al., also only identifies person, location, and organization, but unlike the approach used by Cherry et. al, it uses Conditional Random Fields (CRF). CRF is a probabilistic framework for labeling and segmenting a sequence of data. It represents the probability of a hidden stage sequence, given the observations. The collected tweets with more than four tokens from the Twitter API are cleaned by removing elements like mentions, hyperlinks, hashtags and symbols. The words are assigned their individual part-of-speech tags from the Stanford NLP library. Word2Vec is used for the word representation.

This approach uses a hybrid method consisting of active learning and self-learning as a semi-supervised learning approach. Initially, the classifier begins with a small set of labeled data. If the confidence for the predicted result of an unlabelled instance is high, then this instance is added to the training data along with its labels. The threshold for adding instances is set manually to prevent the addition of noisy data. To select the initial sample of labelled data, instances that provide the most information are selected using AL sampling strategies such as uncertainty-based sampling, diversity-based sampling, and context-based sampling.

This experiment is evaluated on a Twitter dataset scraped using a Twitter API. For the baseline, the precision was 81.6, recall 53.4 and a F1 score of 64.55. After setting the threshold from 0.3 to 0.8, and adjusting for other parameters, the precision was 74.5, recall 66.7 and a F1 score of 70.38 [17].

Across all these papers, some key ideas stand out. From [13] and [16], we can see as expected, when the models are trained on the domain they are labelling they have a better performance. While all the approaches used Word2Vec, from [17], we can see that using Brown clusters in tandem can also lead to an improvement. While CRF is one of the most effective approaches for Twitter NER, neural networks have also proven to be effective as they don't rely on specially crafted features. The addition of active self-learning is important as it helps reduce the human effort for labelling, and also helps the model's performance improve [14].

3 Conclusion and Future Work

3.1 Research Challenges and Future Work

One of the research challenges that we would like to see explored more is using real-world data to augment NER tasks, when there is a lack of unannotated and annotated text data. [9] used Shogi game state data to augment NER on Shogi game commentary. However the F1 score increase was only 0.64%. Thus would like to see more research being done to see if using real data can have more of a significant increase in F1 score. We would also like to see use of more complex and readily-available real-data than game states, such as images and video. There also doesn't exist a common baseline

on a common dataset for NER problems, especially in the field of noisy and short text, that would enable better comparison of different methods.

The lack of labeled data in NER is still a major research challenge. The potential future research directions include the exploration of novel semi-supervised deep learning approaches. Mixup with its foundations in the Vicinal Risk Minimization principle is a very promising research direction with the LADA model in [6] showing empirical evidence of this by its high performance on the semi-supervised NER problem.

3.2 What Value Does This Survey Add

Previous surveys, such as [1] focus on comparing the different deep learning techniques for general NER tasks. However, this survey adds the value of exploring where NER is today in terms of its application in specific domains, some of the practical challenges faced in this process, as well as solutions that have been explored in various papers. Exploring NER domain-level applications is important in getting closer to making enterprise-level NER software.

3.3 Team Responsibilities

Each team member explored a different subtopic of NER, and researched 5 papers in that subtopic. Sanmesh worked on exploring where NER is today in terms of its application in specific domains, some of the challenges faced in this process, as well as solutions that have been explored in various papers. Tanvi worked on investigating how NER is applied to Twitter posts, where the text is noisy and short. Aaron worked on investigating and comparing semi-supervised deep learning approaches to limited labeled data in NER. Apart from working on these sections, all the team members contributed equally to the introduction, and conclusion sections, as well as the overall proofreading of the survey.

References

[1] J. Li, A. Sun, J. Han, and C. Li. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge Data Engineering*, (01):1–1, mar 5555.

[2] Chan Song, Dawn Lawrie, Tim Finin, and James Mayfield. Improving neural named entity recognition with gazetteers, 03 2020.

[3] Hongyi Zhang, M. Cissé, Yann Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *ArXiv*, abs/1710.09412, 2018.

[4] Mingda Chen, Qingming Tang, Karen Livescu, and Kevin Gimpel. Variational sequential labelers for semi-supervised learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 215–226, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[5] Pooja Lakshmi Narayan, Ajay Nagesh, and Mihai Surdeanu. Exploration of noise strategies in semi-supervised named entity classification. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 186–191, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[6] Jiaao Chen, Zhenghui Wang, Ran Tian, Zichao Yang, and Diyi Yang. Local additivity based data augmentation for semi-supervised ner, 2020.

[7] S. Silvestri, F. Gargiulo, and M. Ciampi. Improving biomedical information extraction with word embeddings trained on closed-domain corpora. In *2019 IEEE Symposium on Computers and Communications (ISCC)*, pages 1129–1134, 2019.

[8] X. Li, X. Wang, R. Zhong, D. Zhong, T. He, X. Hu, and X. Jiang. A hybrid deep learning framework for bacterial named entity recognition. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 428–433, 2018.

[9] Suzushi Tomori, Takashi Ninomiya, and Shinsuke Mori. Domain specific named entity recognition referring to the real world by deep neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 236–242, Berlin, Germany, August 2016. Association for Computational Linguistics.

[10] Amogh Kamat Tarcar, Aashis Tiwari, Vineet Naique Dhaimodker, Penjo Rebelo, Rahul Desai, and Dattaraj Rao. Healthcare ner models using language model pretraining, 2020.

[11] Philip John Gorinski, Honghan Wu, Claire Grover, Richard Tobin, Conn Talbot, Heather Whalley, Cathie Sudlow, William Whiteley, and Beatrice Alex. Named entity recognition for electronic health records: A comparison of rule-based and machine learning approaches, 2019.

[12] Rong Li Gai, Fei Gao, Li Ming Duan, Xiao Hui Sun, and Hong Zheng Li. Bidirectional maximal matching word segmentation algorithm with rules. In *Progress in Applied Sciences, Engineering and Technology*, volume 926 of *Advanced Materials Research*, pages 3368–3372. Trans Tech Publications Ltd, 7 2014.

[13] Chenliang Li, Jianshu Weng, Qi He, Yuxia Yao, Anwitaman Datta, Aixin Sun, and Bu-Sung Lee. Twiner: Named entity recognition in targeted twitter stream. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, page 721–730, New York, NY, USA, 2012. Association for Computing Machinery.

[14] Nut Limsopatham and Nigel Collier. Bidirectional lstm for named entity recognition in twitter messages. 2016.

[15] Baptist Vandersmissen Wesley De Neve Godin, Fréderic and Rik Van de Walle. Multimedia lab@ acl wnut ner shared task: Named entity recognition for twitter microposts using distributed word representations. In *Proceedings of the workshop on noisy user-generated text*, pages 146–153. Association for Computational Linguistics, 2015.

[16] Colin Cherry and Hongyu Guo. The unreasonable effectiveness of word representations for twitter named entity recognition. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 735–745, 2015.

[17] Van Cuong Tran, Ngoc Thanh Nguyen, Hamido Fujita, Dinh Tuyen Hoang, and Dosam Hwang. A combination of active learning and self-learning for named entity recognition on twitter using conditional random fields. *Knowledge-Based Systems*, 132:179–187, 2017.

A Supplementary Figures

Label
CHEMICAL
DISEASE
SYMPTOM
DOSAGE

Tag	Meaning
Hu	Human
Tu	Turn
Po	Position
Pi	Piece
Ps	Piece specifier
Mc	Move compliment
Pa	Piece attribute
Pq	Piece quantity
Re	Region
Ph	Phase
St	Strategy
Ca	Castle
Me	Move eval.
Mn	Move name
Ee	Eval. element
Ev	Evaluation
Ti	Time
Ac	Player action
Ap	Piece action
Ao	Other action
Ot	Other notion

Entity Type
ischaemic stroke
haemorrhagic stroke
stroke
glioma tumour
meningioma tumour
metastasis tumour
tumour
subdural haematoma
small vessel disease
atrophy
microhaemorrhage
subarachnoid haemorrhage
haemorrhagic transformation
location:cortical
location:deep
time:old
time:recent

Figure 1: Examples of Named Entity tags in different domains. Left-most tag-set from [10] shows simple tags specialized for Electronic Health records. Middle tag-set from [9] shows more specialized tags for Shogi commentary, that have less probability of existing in general corpus. Right-most tag-set from [11] shows most NE tags that are the most specialized of the three tag sets, and has very little probability of appearing in general annotated corpus

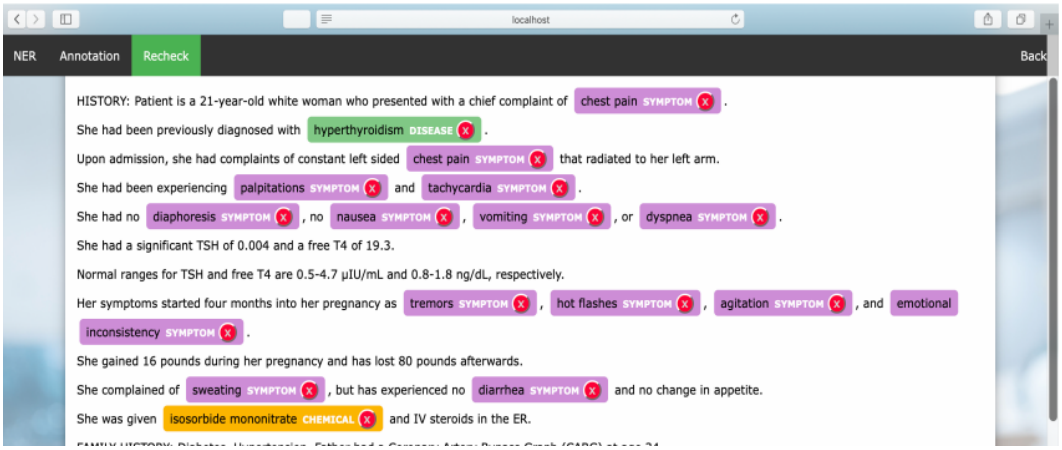

Figure 2: Annotation Tool for domain-specific NER created in [10] in order to speed up the manual annotation process