Learning to recognise named entities in tweets by exploiting weakly labelled data

Kurt Junshean Espinosa[1,2] **Riza Batista-Navarro**[1] **Sophia Ananiadou**[1]
[1]School of Computer Science, University of Manchester, United Kingdom
[2]Department of Computer Science, University of the Philippines Cebu, Philippines
kurtjunshean.espinosa@postgrad.manchester.ac.uk
{riza.batista, sophia.ananiadou}@manchester.ac.uk

Abstract

Named entity recognition (NER) in social media (e.g., Twitter) is a challenging task due to the noisy nature of text. As part of our participation in the W-NUT 2016 Named Entity Recognition Shared Task, we proposed an unsupervised learning approach using deep neural networks and leverage a knowledge base (i.e., DBpedia) to bootstrap sparse entity types with weakly labelled data. To further boost the performance, we employed a more sophisticated tagging scheme and applied dropout as a regularisation technique in order to reduce overfitting. Even without hand-crafting linguistic features nor leveraging any of the W-NUT-provided gazetteers, we obtained robust performance with our approach, which ranked third amongst all shared task participants according to the official evaluation on a gold standard named entity-annotated corpus of 3,856 tweets.

1 Introduction

Named entity recognition (NER) is one of the most fundamental natural language processing (NLP) tasks that is central to understanding unstructured, textual data. Popular approaches (Chieu and Ng, 2002; Florian et al., 2003; Ratinov and Roth, 2009; Luo et al., 2015) mainly rely on hand-crafted features and gazetteers which require knowledge about the domain. Recently, there has been a surge in terms of interest in applying deep learning techniques to NLP tasks. These methods, together with substantial amount of annotated data, can learn features automatically and have been reported to outperform traditional methods on certain NLP tasks (Ma and Hovy, 2016; Lample et al., 2016; Chiu and Nichols, 2016). However, in spite of the perceived success of deep learning methods particularly in NER in newswire (Ma and Hovy, 2016), performance on social media content, particularly that from Twitter, has been lagging behind (Baldwin et al., 2015).

There are two state-of-the-art deep learning methods for newswire NER, namely, those proposed by Chiu and Nichols (2016) and Ma and Hovy (2016). Considering that the former requires hand-crafted features (e.g., word matches against gazetteers) and yet obtains only a small boost in performance over the latter which does not rely on any such features, we took the approach of Ma and Hovy (2016) and applied it on microblog posts from the Twitter platform, i.e., tweets. Based on the results of our initial experiments, we observed suboptimal performance relative to that on newswire. Informed by the error analysis that we carried out, we employed a distant supervision method exploiting an external lexical resource, i.e., DBpedia (Daiber et al., 2013), in order to generate weakly labelled data for low-frequency named entity types. We also investigated the effect of using different token-level tagging schemes and confirmed that improved performance can be obtained by applying the finer-grained BIOES scheme rather than the more popularly used BIO convention. Furthermore, we explored the use of generic placeholders to address out-of-embedding-vocabulary (OOEV) words, although this approach did not lead to better performance.

2 Related Work

Named entity recognition—the automatic demarcation of named mentions within text and their classification according to predefined semantic types—is a fundamental NLP task that has attracted the attention

Proceedings of the 2nd Workshop on Noisy User-generated Text,
pages 153–163, Osaka, Japan, December 11 2016.

of many researchers. Several efforts (Chieu and Ng, 2002; Florian et al., 2003; Ratinov and Roth, 2009), tackle this problem by hand-crafting linguistic features and presenting them to a machine learning algorithm, which will then discriminate between various semantic types based on those features. This particular approach, however, is usually done ad hoc making it very tedious to adapt features to other domains. Recently, extensions to this approach have been introduced (Godin et al., 2015; Cherry and Guo, 2015; Toh et al., 2015) by considering word embeddings as sources of features for classification. Other studies (Luo et al., 2015; Yamada123 et al., 2015) combined this machine learning-based approach with entity linking methods which exploit knowledge bases (e.g., Wikipedia) to detect named mentions. With the NLP research community's continuously surging interest in deep learning, approaches based on deep neural networks have also been applied to NER. While they take away the burden that comes with hand-crafting linguistic features, they require huge amounts of data. Nevertheless, they have been proven to be effective for the NER task (Godin et al., 2015; Chiu and Nichols, 2016; Ma and Hovy, 2016; Lample et al., 2016). For instance, state-of-the-art NER systems for newswire were built upon deep learning-based approaches (Ma and Hovy, 2016; Chiu and Nichols, 2016). It has been shown, however, that even such state-of-the-art methods tend to underperform when applied to other domains, particularly on social media content, e.g., tweets (Baldwin et al., 2015). Challenges in this domain include noise inherent in the data, the many new terms—neologisms—introduced in social media over time, and changes in linguistic conventions in general, also known as language drift (Dredze et al., 2010; Ritter et al., 2011; Eisenstein, 2013; Fromreide et al., 2014).

In this paper, we describe our methods addressing the above-mentioned challenges in the context of the 2016 Named Entity Recognition in Twitter Task (Strauss et al., 2016) organised as part of the Second Workshop on Noisy User-generated Text (W-NUT). For our contribution to the said shared task, we combined deep learning-based approaches with distant supervision methods for generating weakly labelled data, and further optimised performance by exploring the use of different tagging schemes and word embeddings.

3 Methodology

We cast the NER task as a sequence labelling problem: every tweet is a sequence of tokens, each of which is automatically assigned a label (or tag) that is indicative of its membership to a semantic type or category. In learning and applying token labels, two different tagging schemes were compared in order to confirm previously reported observations that employing more sophisticated tagging schemes leads to better predictions (Ratinov and Roth, 2009; Dai et al., 2015). According to the popular begin-inside-outside (BIO) scheme, each token is tagged as any of 'B', 'I' or 'O' depending on whether it is at the *beginning*, *inside* or *outside* a named entity, respectively. The more fine-grained BIOES scheme, however, additionally makes use of 'E' and 'S' to also distinguish tokens at the *end* and those comprising *single-token* entities.

Approaches to sequence labelling based on the conditional random fields (CRF) algorithm are known to demonstrate strong performance (McCallum and Li, 2003; Leaman et al., 2008; Finkel and Manning, 2009). CRFs find the most probable label sequence given a sequence of tokens encoded using features, e.g., surface forms, lemma, part-of-speech tags, surrounding tokens, morphology (Sang and Veenstra, 1999). Meanwhile, convolutional neural networks (CNNs) are sparse feed-forward neural networks which have been shown to effectively extract morphological features such as word prefixes and suffixes (Chiu and Nichols, 2016). One can thus explore the combination of the ability of CRFs to model relations between labels of tokens in a tweet, and the effectiveness of CNNs to extract morphological features of words. CNNs, however, suffer from one drawback, i.e., failure to capture the contextual information surrounding a word. To alleviate this issue, we instead employed recurrent neural networks (RNNs) (Rumelhart et al., 1988). RNN models make use of the sequential information found in structured input (e.g., in a sentence). It is called *recurrent* because it performs the same computation for every element in a sequence with inputs as the only difference. RNN models use the backpropagation algorithm throughout all of the time steps (i.e., from the current word to the *n*th preceding word). During backpropagation, the gradient is computed, which is the measure of how much the cost (i.e., the difference between the expected and the actual value) changes with a change in weight or bias value. The gradient is computed from the output layer (i.e., the current word) and is propagated back to the first layer (i.e., the *n*th preceding word). Since the gradient at a particular point (e.g., current/output layer) is a product of the gradients up to that point (e.g., from the first up to the current layer), the product becomes much smaller or bigger depending on the gradient value. This is called the vanishing/exploding gradient problem common in deep backpropagation investigated more deeply in Bengio et al. (1994) and Pascanu et al. (2013). To remedy this problem, we use a variant of RNNs known as the Long-Short-Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997). LSTM addresses the shortcomings of vanilla RNNs by allowing the network to capture information for an extended number of time steps. Furthermore, we incorporate the findings of Dyer et al. (2015) that bi-directional LSTM (BLSTM) provides more benefits since it allows access to both the past and future contexts, i.e., *m* words to each of the left and right sides of a given word, where *m* is the window size.

3.1 Neural Network Architecture

The input to the network is a concatenation of character representations of words extracted from a convolutional neural network and word embeddings, which will be described in Section 3.3. The inputs are fed into a bi-directional LSTM (BLSTM) and the outputs are concatenated and fed into the CRF layer to jointly decode the best label sequence. For a schematic diagram depicting this neural network architecture, we refer the reader to the work by Ma and Hovy (2016), who first proposed this method for NER in newswire. In our work, the same architecture is applied to the task of NER in tweets.

We also sought to tune the hyperparameters of the neural network, specifically, learning rate and dropout. However, in the interest of time, only a local search within a small interval was carried out, setting the other parameters to arbitrarily determined fixed values. Learning rate controls the step size in gradient descent optimisation: a very small learning rate could lead to longer training time, and potentially to being trapped in a local minimum. Meanwhile, choosing too big a learning rate could lead to overshooting the global optimum. Hence, one way to determine the optimal learning rate is to observe the performance based on a range of different learning rates and selecting the one yielding the maximum performance. Meanwhile, tuning the other important hyperparameter, dropout, has been found to be useful to prevent overfitting. Serving as a means for regularisation, dropout disables units in the network (along with their connections) during training. It is based on a value *p* which indicates the probability that a unit is retained, typically determined using a validation data set. Dropout thus reduces the problem of overfitting and at the same time, provides a way for exploring an exponential number of different neural network architectures efficiently (Srivastava et al., 2014).

3.2 Distant supervision for augmenting training data

We carried out an analysis of the corpus that was provided by the task organisers to participants containing 2,394 and 1,000 tweets in the training and development sets, respectively. The training set is composed of tweets collected in 2010 while those in the development set were collected between 2014 and 2015 (Baldwin et al., 2015). Examining the training set, we observed that whilst 3.6% of the tweets are retweets, 74.2% of the tokens are unique. This same pattern was also observed in the development set: it contains a similar proportion of unique tokens (74.6%), although with much fewer retweets (0.4%). Prepositions, punctuations, articles and pronouns account for many of the repeated tokens in both data sets.

In terms of named entity type distribution, around 96% in each data set correspond to non-named entity tokens. The remaining 4% were distributed unevenly across the 10 entity categories. Based on our frequency analysis, shown in Table 1, we observed that the *person* entity type is the most prevalent in both the training set (449 out of 1,496 entities or 30%) and development set (171 out of 661 entities or 26%) while both *tvshow* and *movie* entity types were consistently the most sparse entities in both the training set (both 34 out of 1491 entities or 2%) and development set (2 out of 661 entities or 0.3%, 15 out of 661 entities or 2%), respectively.

To mitigate the data sparsity of some entity types as observed in the data set, we sought to increase their number of annotated samples. To this end, we leveraged DBpedia Spotlight (Daiber et al., 2013), a

Training Set (1,496 entities)			Development Set (661 entities)		
Entity Type	**Count**	**Percentage**	**Entity Type**	**Count**	**Percentage**
person	449	30.01	**person**	171	25.87
geo-loc	276	18.45	other	132	19.97
other	225	15.04	geo-loc	116	17.55
company	171	11.43	sportsteam	70	10.59
facility	104	6.95	musicartist	41	6.20
product	97	6.48	company	39	5.90
musicartist	55	3.68	facility	38	5.75
sportsteam	51	3.41	product	37	5.60
movie	34	2.27	**movie**	15	2.27
tvshow	34	2.27	**tvshow**	2	0.30

Table 1: Entity type distribution in training and development sets

Entity Type	DBpedia Categories
company	Website, Company, Software, BroadcastNetwork, RadioStation
facility	EducationInstitution, InternationalOrganisation, PublicTransitSystem, ArchitecturalStructure, SportsClub
geo-loc	PopulatedPlace, WorldHeritageSite
other	Non-ProfitOrganisation, Event, SportsLeague, PoliticalParty, MilitaryUnit
movie	Film
musicartist	MusicArtist
person	Person, Athlete, Celebrity, Cleric, Coach, BeautyQueen, BusinessPerson, Criminal, Economist, Engineer, FictionalCharacter, Journalist, Judge, Lawyer, Model, Monarch, MovieDirector, Noble, OrganisationMember, Politician, PoliticianSpouse, Royalty, Scientist, SportsManager, TelevisionDirector, TelevisionPersonality, Writer, OfficeHolder
product	VideoGame, Device, MeanOfTransportation, Food, LineOfFashion, ProgrammingLanguage
tvshow	TelevisionShow, TelevisionSeason, TelevisionEpisode
sportsteam	SportsTeam

Table 2: Proposed mapping between W-NUT entity types and DBpedia categories

tool for linking mentions in text to DBpedia entries, which in our case was employed to weakly annotate our own collection of tweets, gathered between July and August 2016. The tool can be configured in order to define the resulting entity matches. For example, there are configuration parameters, e.g., confidence and contextual score, which can be specified to control the quality of the output (Mendes et al., 2011). Confidence is a parameter for disambiguation, ranging from 0 to 1, which takes into account factors such as topical pertinence and contextual ambiguity. The confidence threshold setting instructs the tool to exclude low-ranking annotations at the risk of losing potentially correct ones. For example, a confidence level of 0.7 will eliminate 70% of low-ranking candidates. Meanwhile, contextual score is the cosine similarity between term vectors weighted by the Term Frequency Inverse Candidate Frequency (TF*ICF) measure introduced by Mendes et al. (2011). ICF is based on the idea that the discriminative power of a word is inversely proportional to the number of DBpedia resources it is associated with. The above parameters provide the tool with a way to rank the list of candidates.

In applying DBpedia Spotlight on our own collection of tweets, we initially set the confidence and contextual score parameters to 0.7. Fuzzy string matching heuristics were also employed, to enable the matching of lexical variants in tweets (e.g., those which differ in length by at most two characters)

Pre-trained Word Embeddings	Description	Text Type
Twitter	2B tweets, 27B tokens, 1.2M vocabulary words, uncased, 100 dimensions	Tweets
Common Crawl	840B tokens, 2.2M vocabulary words, cased, 300 dimensions	Web Pages
Wikipedia 2014 + Gigaword 5	6B tokens, 400K vocabulary words, uncased, 100 dimensions	Wiki pages + newswire

Table 3: Description of word embeddings pre-trained using GloVe

against DBpedia entries. The tool assigns matched entities with semantic labels that come from DBpedia's ontology, which defines a hierarchy of types. In order to categorise these entities according to the shared task's categories of interest, we defined the mappings shown in Table 2, informed by the annotation guidelines provided by the shared task organisers. For many entities, DBpedia Spotlight provides multiple hierarchical labels. For example, the entity "Justin Bieber" is labelled by the tool as both *Person* and *MusicArtist*, which map to W-NUT's *person* and *musicartist*, respectively. In such cases, we take the most specific entity type (e.g., *musicartist*) as the final label. Furthermore, if an entity match x is subsumed by another match y, we disregard x and retain the longer matching entity y with its corresponding type. For example, if a tweet contains three entity matches such as "New York USA", "New York" and "USA", we disregard the two shorter entity matches ("New York", "USA") and choose the entity with the longer span of tokens ("New York USA").

The above techniques, however, did not result in better performance. With the threshold of 0.7, which turned out to be too low, even named entities that DBpedia Spotlight was not very confident about were included in the training set, thus introducing noise. This corresponds to token sequences which were recognised as entities of a certain type when in reality they are actually non-entities or fall under a different entity type. This noise is compounded as the bi-directional LSTM models we employed made use of the context surrounding the wrongly recognised entities to learn the segmentation and classification tasks. To resolve these issues, we increased the required level of confidence in weakly annotated named entity types, from 0.7 to 0.9. Furthermore, we added weakly annotated data only to the entity types for which our models have obtained poor performance based on our initial experiment, namely: *product, movie, musicartist, tvshow*. In this way, we avoid adding unexpected noise with respect to the other named entity types.

3.3 Exploiting different types of word embeddings

Word embeddings are n-dimensional continuous-valued representations of a word where n is an arbitrarily chosen number of dimensions. In contrast to one-hot representations whose number of dimensions would be equal to the vocabulary size V, distributed representations such as word embeddings represent a word in n dimensions where n is much smaller than V (i.e., $n \ll V$). Aside from avoiding sparse representations, word embeddings are shown to capture a word's semantic and syntactic information (Mikolov et al., 2013). Specifically, since word embeddings are generated based on co-occurrence information, co-occurring words share the same context and are thus expected to have similar values.

There are a number of implementations for generating word embeddings such as word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). Pennington et al. (2014) have shown that GloVe produces better results over word2vec (i.e., using CBOW and Skip-gram models) on word similarity and named entity recognition tasks. They also provided pre-trained word embeddings generated based on three different corpora: Common Crawl, Twitter, and Wikipedia 2014+Gigaword 5. Based on the comparison presented in Table 3, it can be observed that Common Crawl has the largest vocabulary size.

Word embeddings have been employed as features for the NER task (Baldwin et al., 2015). As our study is based on unsupervised learning, i.e., neural networks, in which the feature set is learned auto-

Word	Wikipedia+Gigaword	Twitter	Common Crawl
IV	64.66 (9,620)	65.88 (9,802)	**70.65 (10,512)**
OOEV	35.34 (5,258)	34.12 (5,076)	29.35 (4,366)

Table 4: Percentage distribution of IV and OOEV words in the pre-trained embeddings

Category	Percentage
@username	39.31
http://url	21.19
words	19.87
#hashtags	12.15
punctuations	5.43
numbers	2.19

Table 5: OOEV word categories and their distribution

matically, we instead use word embeddings as representations of input words. In this paper, we observe the effect on NER performance, of using each of the three different pre-trained word embeddings.

3.4 Representing Out-of-Embedding Vocabulary (OOEV) Words

Words for which there are no embeddings, i.e., out-of-embedding vocabulary (OOEV) words, commonly occur in the given data set, accounting for around 30% of all words. Table 4 presents the frequency of words with embeddings, i.e., in-vocabulary (IV) words, and OOEV words in the different pre-trained embeddings. We analysed the most frequently occurring OOEV words and categorised them, as shown in Table 5. We observed that the most prevalent type of OOEV words correspond to Twitter usernames of the pattern "@username" as well as URLs, comprising around 60% of the words with no embedding values. Out of the OOEV words, only 10% are named entities, with the rest accounting for context words. Table 6 shows the percentage of named entities with word embeddings (i.e., IV words) and the percentage OOEV words in the training and test sets, respectively.

One way of representing OOEV words is by randomly initialising embeddings with values uniformly sampled from a specified range $[-\sqrt{3/dim}, +\sqrt{3/dim}]$ where *dim* is the dimension of embeddings (He et al., 2015). In this work, we compare that method with the use of placeholder values to replace OOEV words, e.g., by substituting rare words with *'UNK'* (Nam et al., 2016). We applied this on the following OOEV categories: usernames, URLs, punctuations, numbers and hashtags. Furthermore, to account for OOEV words which are potentially lexical variants of IV words (e.g., *'gud'* vs *'good'*, *'#Twitter'* vs *'Twitter'*), we employed string similarity measures based on the edit distance metric (Contractor et al., 2010) to find the IV word most similar to a given OOEV word. This technique was applied on hashtags, punctuations and words. Whilst GloVe comes with its own script for normalisation[1], we chose to implement our own in order to incorporate string similarity measures.

4 Results and Discussion

In this section, we present the results of our evaluation of the methods described above. We investigated the impact on performance of employing different tagging schemes and word embeddings, as well as the substitution of OOEV words with placeholders. We then analysed the extent to which the use of different hyperparameter values and weakly labelled named entities, contributed to NER performance.

As a first step, a decision was made on the data sets to be used for training our models and subsequently evaluating performance. Noting that the shared task-provided training and development data came from disjoint time periods, i.e., 2010 and 2014-2015 respectively, we were interested in determining whether the use of a more temporally heterogeneous set of tweets for training our models could lead to better results. To this end, we formed our own training and development data partitions out of the original data

[1] http://nlp.stanford.edu/projects/glove/preprocess-twitter.rb

Data Set	Total Named Entities	IV	OOEV
Training Set	80.83 (2,902)	93.34 (2,709)	**6.06 (193)**
Test Set	19.17 (688)	95.49 (657)	**4.5 (31)**

Table 6: Percentage distribution of named entities with IV and OOEV words

Tagging Scheme	Precision	Recall	F-score
BIO	**61.76**	53.17	57.14
BIOES	60.99	**57.25**	**59.06**

Table 7: Comparison of performance when using different tagging schemes

sets. Firstly, we combined the tweets from the provided training and development sets. The resulting collection was then partitioned into two: 80% of the tweets became part of a new training set while the remaining 20% formed a new development set. We then ran an experiment to compare the performance obtained when using this new partition, against using the original training and development sets. Our results show that utilising our own data partition yields better performance. This confirms previous observations by Fromreide et al. (2014) that language drift is prevalent in tweets: models trained on relatively older tweets (i.e., from 2010) perform suboptimally on more recent ones (i.e., from 2014-2015). Our own training and development data partitions were thus used in the rest of the experiments described below, together with the following hyperparameter values for the neural network: dropout = 0.6, learning = 0.013 and number of epochs = 63. The value for each hyperparameter was determined by comparing performance across a small number of options: $\{0.3, 0.4, 0.5, 0.6, 0.7\}$ for dropout, $\{0.01, 0.013, 0.015, 0.017, 0.02\}$ for learning rate and values within the interval [1, 1500] for number of epochs.

Table 7 compares the performance obtained by employing the BIO tagging convention versus that using the finer-grained BIOES tagging scheme. As in previous studies (Ratinov and Roth, 2009; Dai et al., 2015), our results show that adopting the BIOES scheme leads to better performance. This can be attributed to the additional information obtained by distinguishing tokens at the end ('E') of entities and those comprising single-token ('S') entities.

We next observed the differences in performance brought about by varying the set of word embeddings (between pre-trained Common Crawl, Twitter and Wikipedia+Gigaword embeddings) used as input to the neural network. It is slightly surprising that optimum performance was obtained with word embeddings learned based on Common Crawl rather than Twitter (Table 8). This, however, can be attributed to Common Crawl's much larger vocabulary (i.e., five times more than Wikipedia+Gigaword and twice the size of the Twitter corpus). We plan to carry out more experiments (as part of future work) to further investigate the effect of exploiting different pre-trained word embeddings.

Contrary to our expectations, embeddings generated by substituting OOEV words with placeholders did not yield better performance compared to randomly initialised embeddings (see Table 9 below). It is highly probable that replacing OOEV words with placeholders weakened the discriminative power of CNNs, which take into consideration character-level features. Similarly, the string similarity-based measures did not have a noticeable impact on performance as the affected words account for only a very small percentage of the OOEV words.

We then evaluated the contribution of augmenting the sparse entity types in the training data with weakly labelled named entities. As presented in Table 10, performance noticeably increased for all en-

GloVe Pretrained Vectors	Precision	Recall	F-score
Twitter	59.83	51.22	55.19
Common Crawl	63.20	54.88	**58.75**
Wikipedia 2014 + Gigaword 5	58.46	46.34	51.70

Table 8: Comparison of performance based on different pre-trained word embeddings

Embedding Strategy	**Precision**	**Recall**	**F-score**
Randomly initialised embeddings (baseline)	63.20	54.88	**58.75**
Randomly initialised embeddings (words) + placeholder(all others)	64.13	51.84	57.34
string sim (words, hashtags, punctuations) + placeholder(url, @username, number)	62.28	51.11	56.14
string sim (words, hashtags) + placeholder(url, @username, number, punctuations)	62.92	50.86	56.25

Table 9: Performance of different embedding strategies for representing OOEV words

Entity Type	**Training Set only**	**Training Set + Weakly Labelled entities for sparse types**
tvshow	44.44	**44.44**
product	40	**41.38**
musicartist	28.57	**46.15**
movie	0	**16.67**

Table 10: Effect on the F-scores after adding weakly labelled data to sparse entity types

tity types except in the case of *tvshow*, in which no improvement was obtained. The effect is especially high for *movie* and *musicartist*—entity types for which the F-scores were previously zero. This demonstrates that the incorporation of weakly labelled data in training the neural network can lead to significant improvement with respect to sparse entity types.

Informed by the above results, we trained our final NER model on our training data partition augmented with weakly labelled named entities, using: (1) the BIOES tagging scheme; (2) pre-trained Common Crawl word embeddings; and (3) randomly initialised word embeddings for OOEV words. Model training took two hours on an Nvidia M2050 GPU processor.

Underpinned by the final NER model, our system ("akora") was applied on the shared task's test data, yielding an overall precision = 51.70, recall = 39.48, and F-score = 44.77. It ranked third amongst participating systems (Strauss et al., 2016) as shown in Table 11. Details of the evaluation results according to entity type are presented in Table 12.

Unfortunately, we have been unable to carry out a performance-wise comparison with the systems that participated in the previous edition of the W-NUT NER task. Their performance results were reported based on a data set that formed part of the development data set that we used to fine-tune our neural network architecture's hyperparameters and optimise our approach's performance. Thus, reporting our system's results on the same data set would not have given a fair comparison.

A model for the W-NUT's named entity segmentation task was trained in the same way as above, but with the entity type labels removed from the training data. When applied on the official test data, this model obtained an F-score of 59.05 (precision = 64.75, recall = 54.28).

System Name	**Precision**	**Recall**	**F-score**
CambridgeLTL	60.77	46.07	52.41
Talos	58.51	38.12	46.16
akora	51.70	39.48	44.77
NTNU	53.19	32.13	40.06
ASU	40.58	37.58	39.02

Table 11: Performance of our NER system, akora, compared to other top-ranking W-NUT 2016 systems

Entity Type	Precision	Recall	F-score
company	67.11	32.53	43.82
facility	48.03	28.85	36.05
geo-loc	61.41	65.31	63.30
movie	10.00	2.94	4.55
musicartist	21.05	4.19	6.99
other	37.23	23.97	29.17
person	47.97	58.71	52.80
product	33.33	12.60	18.29
sportsteam	38.89	38.10	38.49
tvshow	10.00	3.03	4.65
Overall	51.70	39.48	**44.77**

Table 12: Named entity recognition performance on the official evaluation data set

5 Conclusion and Future Work

Our experiments have shown that taking advantage of weakly annotated data to alleviate data sparsity of some named entity types resulted in increased performance in the respective entity types. In contrast, mitigating OOEV words by using placeholders did not result in better performance. Future work includes performing further error analysis by means of hidden layer visualisation to enable us to investigate representations created at each layer. Moreover, Bayesian optimisation of parameters can be done as proposed by Rasmussen (2006). Finally, considering the language drift inherent in social media content, looking into word sense disambiguation may help improve performance.

Acknowledgement

We thank our anonymous reviewers for their invaluable feedback, as well as Piotr Przybyła for his comments on the final draft. The first author gratefully acknowledges financial support from the University of the Philippines System Doctoral Studies Fund.

References

Timothy Baldwin, Young-Bum Kim, Marie Catherine de Marneffe, Alan Ritter, Bo Han, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. *ACL-IJCNLP*, 126:2015.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Colin Cherry and Hongyu Guo. 2015. The unreasonable effectiveness of word representations for Twitter named entity recognition. In *Proc. NAACL*.

Hai Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: a maximum entropy approach using global information. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Danish Contractor, Tanveer A Faruquie, and L Venkata Subramaniam. 2010. Unsupervised cleansing of noisy text. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 189–196. Association for Computational Linguistics.

Hong-Jie Dai, Po-Ting Lai, Yung-Chun Chang, and Richard Tzong-Han Tsai. 2015. Enhancing of chemical compound and drug name recognition using representative tag scheme and fine-grained tokenization. *Journal of cheminformatics*, 7(1):1.

Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. 2013. Improving efficiency and accuracy in multilingual entity extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*.

Mark Dredze, Tim Oates, and Christine Piatko. 2010. We're not in Kansas anymore: detecting domain changes in streams. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 585–595. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *HLT-NAACL*, pages 359–369.

Jenny Rose Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 326–334. Association for Computational Linguistics.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 168–171. Association for Computational Linguistics.

Hege Fromreide, Dirk Hovy, and Anders Sgaard. 2014. Crowdsourcing and annotating NER for Twitter# drift. In *LREC*, pages 2544–2547.

Frderic Godin, Baptist Vandersmissen, Wesley De Neve, and Rik Van de Walle. 2015. Multimedia Lab@ ACL W-NUT NER Shared Task: Named Entity Recognition for Twitter Microposts using Distributed Word Representations. *ACL-IJCNLP 2015*, page 146.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

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

Robert Leaman, Graciela Gonzalez, and others. 2008. BANNER: an executable survey of advances in biomedical named entity recognition. In *Pacific symposium on biocomputing*, volume 13, pages 652–663.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint named entity recognition and disambiguation. In *Proc. EMNLP*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. *arXiv preprint arXiv:1603.01354*.

Andrew McCallum and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 188–191. Association for Computational Linguistics.

Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. 2011. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8. ACM.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jinseok Nam, Eneldo Loza Menca, and Johannes Frnkranz. 2016. All-in Text: Learning Document, Label, and Word Representations Jointly. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43.

Carl Edward Rasmussen. 2006. Gaussian processes for machine learning. MIT Press.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Alan Ritter, Sam Clark, Oren Etzioni, and others. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

Erik F Sang and Jorn Veenstra. 1999. Representing text chunks. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*, pages 173–179. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Benjamin Strauss, Bethany E. Toma, Alan Ritter, Marie Catherine de Marneffe, and Wei Xu. 2016. Results of the wnut16 named entity recognition shared task. In *Proceedings of the Workshop on Noisy User-generated Text (WNUT 2016)*, Osaka, Japan.

Zhiqiang Toh, Bin Chen, and Jian Su. 2015. Improving twitter named entity recognition using word representations.

Ikuya Yamada123, Hideaki Takeda, and Yoshiyasu Takefuji. 2015. Enhancing Named Entity Recognition in Twitter Messages Using Entity Linking. *ACL-IJCNLP 2015*, page 136.