NRC: Infused Phrase Vectors for Named Entity Recognition in Twitter

Colin Cherry and **Hongyu Guo** and **Chengbi Dai**
National Research Council Canada
first.last@nrc-cnrc.gc.ca

Abstract

Our submission to the W-NUT Named Entity Recognition in Twitter task closely follows the approach detailed by Cherry and Guo (2015), who use a discriminative, semi-Markov tagger, augmented with multiple word representations. We enhance this approach with updated gazetteers, and with infused phrase embeddings that have been adapted to better predict the gazetteer membership of each phrase. Our system achieves a typed F1 of 44.7, resulting in a third-place finish, despite training only on the official training set. A post-competition analysis indicates that also training on the provided development data improves our performance to 54.2 F1.

1 Introduction

Named entity recognition (NER) is the task of finding rigid designators as they appear in free text and assigning them to coarse types such as *person* or *geo-location* (Nadeau and Sekine, 2007). NER is the first step in many information extraction tasks, but in social media, this task is extremely challenging. The text to be analyzed is unedited and noisy, and covers a much more diverse set of topics than one might expect in newswire. As such, we are quite interested in the W-NUT Named Entity Recognition in Twitter task (Baldwin et al., 2015) as a platform to benchmark and drive forward work on NER in social media.

Our submission to this competition closely follows Cherry and Guo (2015), who advocate the use of a semi-Markov tagger trained online with standard discriminative tagging features, gazetteer matches, Brown clusters, and word embeddings. We augment this approach with updated gazetteers, phrase embeddings, and *infused embeddings* that have been adapted to better predict gazetteer membership. Our novel infusion technique allows us to adapt existing vectors to NER regardless of their source, by training a type-level auto-encoder whose hidden layer must predict the corresponding phrase's gazetteer memberships while also recovering the original vector.

Our submitted system achieved a typed F1 of 44.7, placing third in the competition, while training only on the provided training data. The competition organizers provided two development sets, one (dev) that is close to the training data, with both train and dev being drawn from the year 2010, and another (dev_2015) that is close to the test data, with both dev_2015 and test being drawn from the winter of 2014–2015. We present a post-competition system that achieves an F1 of 54.2 using the same features and hyper-parameters as our submitted system, except that our tagger is also trained on all provided development data. We close with an analysis of dev_2015's relation to the test set, and argue that these results may overestimate the impact that a small, in-domain training set can have on NER performance.

2 Data Resources

We make use of two external data resources: gazetteers and unlabeled tweets. For gazetteers, we begin with the word lists provided with the W-NUT baseline system, which appear to be mostly derived from Freebase. We treat each file in the lexicon directory as a distinct word list. We update and augment these lists with our own Freebase queries in Section 3.2.

We use unannotated tweets to build various word representations (see Section 3.1). Our unannotated corpus collects 98M tweets (1,995M tokens) from between May 2011 and April 2012. The same corpus is used by Cherry and Guo (2015). These tweets have been tokenized and post-processed to remove many special Unicode

Proceedings of the ACL 2015 Workshop on Noisy User-generated Text, pages 54–60,
Beijing, China, July 31, 2015. ©2015 Association for Computational Linguistics

characters; they closely resemble those that appear in the provided training and development sets. Furthermore, the corpus consists only of tweets in which the NER system of Ritter et al. (2011) detects at least one entity. The automatic NER tags are used only to select tweets for inclusion in the corpus, after which the annotations are discarded. Filtering our tweets in this way has two immediate effects: first, each tweet is very likely to contain an entity mention. Second, the tweets are very long, with an average of 20.4 tokens per Tweet.

As the test data is drawn from the winter of 2014–2015, we attempted to augment our corpus with more recent data: 13M unannotated English tweets drawn from Twitter's public stream, from between April 24 and May 6, 2015. As we had very little recent data, we made no attempt to bias the corpus to be entity-rich. This corpus of recent tweets has an average tweet length of only 13.8 tokens. Our attempts to use this data to build word representations did not improve NER performance on the 2015 development set, regardless of whether we used the data on its own or in combination with our larger corpus.

3 Methods

3.1 Base Tagger

We first summarize the approach of Cherry and Guo (2015), which we build upon for our system.

Tagger: We tag each tweet independently using a semi-Markov tagger (Sarawagi and Cohen, 2004), which tags phrasal entities using a single operation, as opposed to traditional word-based entity tagging schemes. An example tag sequence, drawn from the 2010 development data, is shown in Figure 1. Semi-Markov tagging gives us the freedom to design features at either the phrase or the word level, while also simplifying our tag set. Furthermore, with our semi-Markov tags, we find we have no need for Markov features that track previous tag assignments, as our entity labels cohere naturally. This speeds up tagging dramatically. Semi-Markov tagging also introduces a hyper-parameter P, the maximum entity length in tokens.

Training: Our tagger is trained online with large-margin updates, following a structured variant of the passive aggressive (PA) algorithm (Crammer et al., 2006). We regularize the model both with a fixed number of epochs E through the data, and using PA's regularization term C, which is similar to that of an SVM. We also have the capacity to deploy example-specific C-parameters, allowing us to assign some examples more weight during training, which we use only in post-competition analysis.

Lexical Features: Recall that our semi-Markov model allows for both word and phrase-level features. The vast majority of our features are word-level, with the representation for a phrase being the sum of the features of its words. Our word-level features closely follow the set proposed by Ratnaparkhi (1996), covering word identity, the identities of surrounding words within a window of 2 tokens, and prefixes and suffixes up to three characters in length. Each word identity feature has three variants, with the first reporting the original word, the second reporting a lowercased version, and the third reporting a summary of the word's shape ("Mrs." becomes "Aa.") All word-level features also have a variant that appends the word's Begin/Inside/Last/Unique position within its entity. Our phrase-level features report phrase identity, with lowercased and word shape variants, along with a bias feature that is always on. Phrase identity features allow us to memorize tags for common phrases explicitly. Following the standard discriminative tagging paradigm, all features have the tag identity appended to them.

Representation Features: We also produce word-level features corresponding to a number of external representations: gazetteer membership, Brown clusters (Brown et al., 1992) and word embeddings. For gazetteers, we first segment the tweet into longest matching gazetteer phrases, resolving overlapping phrases with a greedy left-to-right walk through the tweet. Each word then generates a set of features indicating which gazetteers (if any) contain its phrase. For cluster representations, we train Brown clusters on our unannotated corpus, using the implementation by Liang (2005) to build 1,000 clusters over types that occur with a minimum frequency of 10. Following Miller et al. (2004), each word generates indicators for bit prefixes of its binary cluster signature, for prefixes of length 2, 4 8 and 12. For word embeddings, we use an in-house Java re-implementation of word2vec (Mikolov et al., 2013a) to build 300-dimensional vector representations for all types that occur at least 10 times in our unannotated corpus. Each word then reports a real-valued feature (as opposed to an indicator) for each of the 300

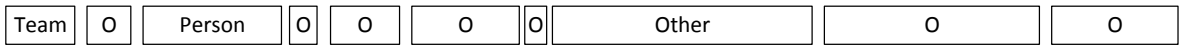

Figure 1: An example of semi-Markov tagging.

dimensions in its vector representation. A single random vector is created to represent all out-of-vocabulary words. Our vectors and clusters cover 2.5 million types. Note that we do not include part-of-speech tags as features, as they were not found to be useful by Cherry and Guo (2015).

3.2 Updated Gazetteers

During development, we found that features involving gazetteers were having a larger impact on dev than on dev_2015. Therefore, we enhanced the gazetteers provided with the W-NUT baseline system with our own Freebase queries, issued between May 6–7, 2015. These updates are summarized in Table 1. The baseline lexicons for which we could infer Freebase categories were replaced with updated queries, indicated with *new=no*. We also added a number of entirely new queries (*new=yes*). Any baseline lexicon that is not mentioned in Table 1 was left untouched, and remains included in our updated gazetteers.

3.3 Phrase Embeddings

The work of Passos et al. (2014) suggests that embeddings built over phrases may be more useful to NER than those built over words. To test this for our tagger, we use the phrase finding tool provided with word2vec to segment our unannotated corpus into phrases up to 4 tokens in length (Mikolov et al., 2013b). Their software uses a simple statistic similar to pointwise mutual information to assess whether two tokens should be combined into a phrase. Token-pairs passing a threshold are segmented into phrases, creating a new corpus. Phrases longer than 2 tokens can be generated by running this process repeatedly, allowing, for example, *Toronto* to merge with *Maple_Leafs* on the second pass. Once the phrasal corpus has been created, we run word2vec as usual. There is no reason the same procedure could not be applied to Brown clustering, only time constraints prevented us from doing so.

The resulting phrase embeddings are used in place or word embeddings, mostly for efficiency considerations. We assign each word in the tweet to its longest embedded phrase that matches the tweet, resolving conflicts with a greedy, left-to-right matching process. Each word is then assigned a vector corresponding to its matched phrase, meaning that the same vector will be repeated for each token in its phrase. This has the property of having words receive different representations, depending on their context. Otherwise, the embedding features are identical to those described in Section 3.1. Phrase embeddings enable more gazetteer matches for gazetteer-infused vectors, which we discuss next.

3.4 Gazetteer-Infused Phrase Vectors

We employ an auto-encoder to leverage knowledge derived from domain-specific gazetteers to make the distributed phrase representations more relevant to our NER task. In recent years, two sources of information have been found to be valuable to boost the performance for NER: distributed representation learned from a large corpus and domain-specific lexicons (Turian et al., 2010; Cherry and Guo, 2015). Research has also shown that merging these two forms of information can result in further predictive improvement for an NER system (Passos et al., 2014). A similar strategy for enhancing word embeddings has also been demonstrated for sentiment analysis (Tang et al., 2014). Following this line of research, we aim to tailor (post-process) the unsupervised phrase embeddings, created in Section 3.3, for our NER task, using an auto-encoder.

The auto-encoder eliminates the need to have access to the original training data and the vector training model, requiring only the trained distributed vectors. In this sense, it can be considered computationally lighter than the above mentioned information fusion methods.[1] Our approach is inspired by Ngiam et al. (2011) and Glorot et al. (2011), where auto-encoders are efficiently deployed to generate improved features for domains or modalities that are different from those of its inputs. Here, we employ an auto-encoder to inject

[1] In practice, a number of popular sets of pre-trained embedding vectors, trained with very large corpora, are made available online to the research community, but without the original training data; such as some sets from http://nlp.stanford.edu/projects/glove/ and https://code.google.com/p/word2vec/

Gazetteer	#	new
architecture.museum	9k	no
architecture.structure	111k	yes
broadcast.tv_channel	1k	no
business.brand	9k	no
business.consumer_company	2k	no
business.consumer_product	439k	no
business.employer	282k	yes
business.sponsor	2k	no
business.company	393k	yes
location.location	1,336k	no
location.citytown*	189k	yes
location.country	1k	no
location.state_province_region*	66k	yes
film.film	262k	yes
music.artist	613k	yes
music.musical_group	195k	yes
people.family_name	6k	no
people.person	3,200k	no
business.product_line	439k	no
sports.professional_sports_team	1k	yes
sports.school_sports_team	2k	yes
sports.sports_facility	7k	yes
sports.sports_league	4k	no
sports.sports_team	33k	no
tv.tv_network	3k	no
tv.tv_program	74k	no
tv.tv_series_episode	1,316k	yes

Table 1: The Freebase gazetteers we either updated (*new=no*) or added (*new=yes*) to the baseline gazetteers. Freebase categories can be recovered by replacing "." with "/". Gazetteers marked with a * were extracted from /location/mailing_address using the indicated property name.

relevant entity type information derived from our gazetteers into the pre-trained phrase representations.

Using learned phrase vectors as input, the auto-encoder's goal is to reconstruct both the provided input vector V and its entity types, as derived from the collected gazetteers. In our experiments, we assume the entity membership vector has a 0-1 encoding. That is, if there are G lexicon types, then it has length G, where a 1 indicates membership in the corresponding gazetteer. We implement the squared error as the reconstruction cost criterion for the auto-encoder training:

$$[V; G] = f(W_d f(W_e V + b_e) + b_d)$$

where f denotes the hyperbolic *tanh* function, while b_e and b_d are the biases for the encoder and decoder, respectively. We initialize the parameters, namely the decoder matrix W_d and encoder matrix W_e, by randomly sampling each value from a uniform distribution [-0.1, +0.1]. Our experiments use a learning rate of 0.001 and a moment of 0.9. We found the optimal size for the hidden encoding layer to be 200 nodes. The auto-encoder is trained using Stochastic Gradient Descent (SGD) with 100 iterations, which converges very well for our data.

With the above experimental settings, our Gazetteer-Infused phrase vectors are created with two stages. During the initial phase, we select the 69,329 phrases that are shared by both the phrase vectors and the collected gazetteers. The resulting set of phrases is a very small fraction of the total of 4.5M vector entries created in Section 3.3. Consequently, the overwhelming number of negative examples causes a highly imbalanced class distribution problem for the gazetteer membership component of our auto-encoder (Guo and Viktor, 2004). To cope with this skewed class challenge, we randomly down-sample the negative training data to balance the negative and positive instances. These rebalanced data are then fed into the auto-encoder to optimize the parameter matrices. In the second stage, the trained auto-encoder is used to generate a new vector $[V; G]$ for every phrase created in Section 3.3. These gazetteer-infused phrase vectors include a decoded version of the original embedding V, as well as explicit soft predictions of gazetteer membership in G. Note that we apply this process for all 45 of our gazetteers, and not just those that correspond directly to the types tagged in this task.

4 Results

We have collected results for our submitted system, along with some salient pre- and post- competition variants in Table 2. We discuss these results in detail below.

4.1 Competition Results

Our submitted system is shown as $[A]+[U]$, and includes all of the features described in Section 3. It achieves our highest results on both dev_2015 and the average of dev and dev_2015, and its performance on test was sufficient to place third in the competition.

System	dev P	dev R	dev F	dev_2015 P	dev_2015 R	dev_2015 F	Avg F	test P	test R	test F	Rnk
Baseline	57.0	44.4	49.9	38.5	30.9	34.3	42.1	35.6	29.1	32.0	-
Our Baseline	64.6	38.5	48.2	47.7	27.2	34.7	41.5	44.4	23.3	30.6	8
C&G 2015	68.6	50.3	58.0	56.4	41.9	48.1	53.1	49.8	36.5	42.1	5
Inc. Regularization	70.7	50.8	59.2	57.2	42.3	48.6	53.9	51.4	36.8	42.9	4
[P]hrase vectors	69.4	50.8	58.7	58.0	42.7	49.2	53.9	52.6	37.8	44.0	3
[A]dapted vectors	70.0	51.7	59.5	59.9	42.3	49.6	54.5	52.2	37.5	43.7	4
[U]pdated gazetteers	68.5	51.4	58.8	57.4	42.7	49.0	53.9	52.0	37.4	43.5	4
[P]+[U]	73.7	54.2	**62.5**	59.7	44.1	50.7	56.6	53.2	38.9	44.9	3
[A]+[U] (**Submitted**)	72.8	53.4	61.6	62.4	44.5	**51.9**	**56.8**	53.2	38.6	44.7	3
+ dev	-	-	-	59.0	44.5	50.7	-	54.9	41.0	46.9	3
+ dev + dev_2015	-	-	-	-	-	-	-	62.5	47.8	**54.2**	2

Table 2: Experimental results for variants of our system, reporting **P**recision, **R**ecall and balanced **F**-measure. The **Avg F** column lists the average F-measure across dev and dev_2015, which was our model selection criterion. The **Rnk** column lists the retro-active rank of each system in the competition.

4.2 Ablation

The systems above [A]+[U] are intended to demonstrate our development process. *Our baseline* is our attempt to re-implement the provided baseline in our code base, and includes all lexical features and the baseline gazetteers.

C&G 2015 adds Brown clusters and word embeddings to create a complete re-implementation of Cherry and Guo (2015). We can see that these representations have a huge impact on NER performance for all dev and test sets.

We then performed a careful hyper-parameter sweep using the two provided development sets, resulting in the *Inc. Regularization* system. The hyper-parameters suggested by Cherry and Guo (2015) (E=10, C=0.01, P=10) were selected to work well with and without representations. We found that once we have committed to using representations, the tagger benefits from increased regularization, so long as we allow the model to converge (E=30, C=0.001, P=8). Although we revisited these settings periodically, these hyper-parameters have proved to be quite stable, and we use them for all remaining experiments.

The next three systems test the three extensions described in Section 3. Neither [P]*hrase vectors* nor [U]*pdated gazetteers* were able to improve both dev and dev_2015 when applied alone, while the [A]*dapted vectors* did boost performance on both sets, increasing average F-measure by 0.6. In particular, the adapted vectors improved the rare entity types such as *movie* and *sports team*. Unfortunately, these improvements do not seem to carry over to the test set.

As we combine the ideas with [P]+[U] and [A]+[U], we see even larger improvements on both development sets. Note that adapted vectors implicitly include phrase vectors, as those are the vectors that have been adapted. These ideas may work better in combination because both our phrase vectors and our updated gazetteers include many noisy phrasal entries, but their sources of noise are independent, allowing one to compensate for the other.

4.3 Post-Competition Results

All systems discussed thus far have been trained only on the official training data. The final two systems in Table 2 test the impact of adding dev (599 tweets from 2010) and dev_2015 (420 tweets from 2014–2015) to the 1,795 training tweets. When training with dev_2015, we take advantage of our system's data-weighting capabilities to assign these examples twice as much weight; this hyper-parameter was selected only based on dev_2015's relatively small size, and we did not test any other values for it. As one can see, both development sets have a significant impact on test performance, with dev_2015 producing a 7.3 point improvement in F-measure, eclipsing the impact of all other enhancements to our system.

We chose not to include dev in our final system because it did not appear to have a positive impact on dev_2015. We chose not to include dev_2015 in our final system because cross validation experiments, where we train including half of dev_2015

and then test on the other half, indicated that its impact would be minimal (less than 1 point of F-measure), and we did not want to discard the safety net that a held-out development set provides when selecting a final system. In retrospect, this was a fairly large mistake.

4.4 Dev-Test Data Analysis

Does this mean that 420 examples drawn from a time period close to that of the test set will consistently provide 7 points of F-measure? We do not believe so, for at least two reasons: time overlap and Twitter bots.

Time overlap: dev_2015 and test appear to have both been drawn from overlapping periods of time. Though the tweets provided for development and test were not dated, we can infer the date spans from various bots that tweet the date, producing tweets like:

> @ABCD http://t.co/MA3WYTR72Q February 02 , 2015 at 10:57 PM

throughout both sets. Using the regular expression "(December|(Jan|Febr)uary) [0-9]+ , 201[4-5] at" we were able to find tweets between December 11, 2014 and February 4, 2015 in dev_2015, and between December 12, 2014 and February 5, 2015 in test. This means that both sets contain the same major holidays, such as Christmas, and sporting events, such as Super Bowl 49. Accordingly, our post-competition system sees its largest improvements in the types *sports team* and *other* (which covers many event names). These sorts of improvements are not necessarily indicative of how a system trained on data from the past will perform on new data, which is a common use case for NER. This also highlights the importance of having an NER system's training data be drawn from throughout the year. The official training data has no mention of the Super Bowl and very few of Christmas, despite these being yearly events.

Twitter bots: bots are common in both dev_2015 and test, but much less prominent in train and dev, perhaps reflecting an overall change in twitter traffic between 2010 and 2015. For the most part, bots are harmless, and a system should be tested in terms of its ability to ignore these sources of noise. However, some bots tweet entities in an extremely formulaic manner, and a discriminative NER system needs to see very few tweets from such a bot in order to tag it consistently. One such example from this competition is the horoscope bot, which produces tweets that look like:

> Your boss may be critical of your easy-going attitude at work t ... More for Cancer http://t.co/74bwPzVbiB

This bot can be detected reliably with the regular expression "[.][.][.] More for [A-Z]". In the gold-standard annotations, tweets from this bot consistently have the astrological sign (Cancer in this case) tagged as *other*. While this bot appears only 4 times in dev_2015, it appears 23 times in test. If we remove these 23 tweets from the test set, our submitted system increases is performance to a Precision / Recall / F-measure of 53.5 / 40.0 / 45.8, while our best post-competition system decreases to 61.0 / 46.2 / 52.6, narrowing the gap in F-measure by 2.6 points. The ability to extract entities from formulaic bots such as this one could be useful, but the core purpose of NER technology is to enable the extraction of information from human-written text.

5 Conclusion

We have summarized our entry to the first W-NUT Named Entity Recognition in Twitter task. Our entry extends the work of Cherry and Guo (2015) with updated lexicons, phrase embeddings, and gazetteer-infused phrase embeddings. Our gazetteer infusion technique is novel in that it allows us to adapt existing vectors, regardless of their source. Taken together with improved hyper-parameters, these extensions improve the approach of Cherry and Guo (2015) by 2.6 F-measure on a completely blind test. Our final submission achieves a test F-measure of 44.7, placing third in the competition, and could have achieved an F-measure of 54.2 had we included all development data as training data. We have also presented a discussion of how the most recent development set relates to the test set, arguing that these results likely over-estimate the impact of a small, in-domain training set on NER performance.

References

Timothy Baldwin, Marie Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text (WNUT 2015)*, Beijing, China.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

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

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, pages 513–520.

Hongyu Guo and Herna L. Viktor. 2004. Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. *SIGKDD Explorations*, 6(1):30–39.

Percy Liang. 2005. *Semi-supervised learning for natural language*. Ph.D. thesis, Massachusetts Institute of Technology.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *ICLR Workshop*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *HLT-NAACL*, pages 337–342.

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

Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. 2011. Multimodal deep learning. In *ICML*, pages 689–696.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *CoNLL*, pages 78–86.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *EMNLP*, pages 133–142.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *EMNLP*, pages 1524–1534, Edinburgh, Scotland, UK.

Sunita Sarawagi and William W Cohen. 2004. Semimarkov conditional random fields for information extraction. In *NIPS*, pages 1185–1192.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL*, pages 1555–1565.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *ACL*, pages 384–394.