Robust Lexical Features for Improved Neural Network Named-Entity Recognition

Abbas Ghaddar
RALI-DIRO
Université de Montréal
Montréal, Canada
abbas.ghaddar@umontreal.ca

Philippe Langlais
RALI-DIRO
Université de Montréal
Montréal, Canada
felipe@iro.umontreal.ca

Abstract

Neural network approaches to Named-Entity Recognition reduce the need for carefully hand-crafted features. While some features do remain in state-of-the-art systems, lexical features have been mostly discarded, with the exception of gazetteers. In this work, we show that this is unfair: lexical features are actually quite useful. We propose to embed words and entity types into a low-dimensional vector space we train from annotated data produced by distant supervision thanks to Wikipedia. From this, we compute — offline — a feature vector representing each word. When used with a vanilla recurrent neural network model, this representation yields substantial improvements. We establish a new state-of-the-art F1 score of 87.95 on ONTONOTES 5.0, while matching state-of-the-art performance with a F1 score of 91.73 on the over-studied CONLL-2003 dataset.

1 Introduction

Named-Entity Recognition (NER) is the task of identifying textual mentions and classifying them into a predefined set of types. Various approaches have been proposed to tackle the task, from hand-crafted feature-based machine learning models like conditional random fields (Finkel et al., 2005) and perceptron (Ratinov and Roth, 2009), to deep neural models (Collobert et al., 2011; Ma and Hovy, 2016; Strubell et al., 2017).

Word representations (Turian et al., 2010; Mikolov et al., 2013), also known as word embeddings, are a key element for multiple NLP tasks including NER (Collobert et al., 2011). Due to the small amount of named-entity annotated data, embeddings are used to extend, rather than replace, hand-crafted features in order to obtain state-of-the-art performance (Lample et al., 2016). Recent studies (Yang et al., 2017; Søgaard and Goldberg, 2016) have explored methods for supplying deep sequential taggers with complementary features to standard embeddings. Peters et al. (2017) and Tran et al. (2017) tested special embeddings extracted from a neural language model (LM) trained on a large corpus. LM embeddings capture context-dependent aspects of word meaning using future (forward LM) and previous (backward LM) context words. When this information is added to standard features, it leads to significant improvements in NER. Also, Chiu and Nichols (2016) showed that external knowledge resources (namely gazetteers) are crucial to NER performance. Gazetteer features encode the presence of word n-grams in predefined lists of NEs.

In this work, we discuss some of the limitations of gazetteer features and propose an alternative lexical representation which is trained offline and that can be added to any neural NER system. In a nutshell, we embed words and entity types into a joint vector space by leveraging WiFiNE (Ghaddar and Langlais, 2018), a ressource which automatically annotates mentions in Wikipedia with 120 entity types. From this vector space, we compute for each word a 120-dimensional vector, where each dimension encodes the similarity of the word with an entity type. We call this vector an LS representation, for Lexical Similarity. When included in a vanilla LSTM-CRF NER model, LS representations lead to significant gains. We

Proceedings of the 27th International Conference on Computational Linguistics, pages 1896–1907
Santa Fe, New Mexico, USA, August 20-26, 2018.

establish a new state-of-the-art F1 score of 87.95 on ONTONOTES 5.0, while matching state-of-the-art performance on the over-studied CoNLL-2003 dataset.

In the rest of this paper, we motivate our work in Section 2. We describe how we compute LS vectors in Section 3. We present our system in Section 4 and report results in Section 5. In Section 6, we discuss related works before concluding in Section 7.

2 Motivation

Gazetteers are lists of entities that are associated with specific NE categories. They are widely used as a feature source in NER, and have been successfully included in feature-based (Ratinov and Roth, 2009) and neural (Chiu and Nichols, 2016) models. Typically, lists of entities are compiled from structured data sources such as DBpedia (Auer et al., 2007) or Freebase (Bollacker et al., 2008). The surface form of the title of a Wikipedia article, as well as aliases and redirects are mapped to an entity type using the `object_type` attribute of the related DBpedia (or Freebase) page. Ratinov and Roth (2009) use this methodology to compile 30 lists of fine-grained entity types extracted from Wikipedia, while Chiu and Nichols (2016) create 4 gazetteers that map to CoNLL categories (PER, LOC, ORG and MISC). Despite their importance, gazetteer-based features suffer from a number of limitations.

- **Binary representation.** Gazetteer features encode only the presence of an n-gram in each list and omit its relative frequency. For example, the word "France" can be used as a person, an organization, or a location, while it likely refers to the country most of the time. Binary features cannot capture this preference.

- **Generation.** At test time, we need to match every n-gram (up to the length of the longest lexicon entry) in a sentence against entries in the lexicons, which is time consuming. In their work, Chiu and Nichols (2016) use 4 lists that count over 2.3M entries.

- **Non-entity words.** Gazetteer features do not capture signal from non-entity words, while earlier feature-based models strived to encode that some words (or n-grams) trigger specific entity types. For instance, words such as "eat", "directed" or "born" are words that typically appear after a mention of type PER.

To overcome those limitations, we propose an alternative approach where we embed annotations mined from Wikipedia into a vector space from which we compute a feature vector that represent words. This vector compactly and efficiently encodes both gazetteer and lexical information. Note that at test time, we only have to feed our model with this feature vector, which is efficient.

3 Our Method

3.1 Embedding Words and Entity Types

Turning Wikipedia into a corpus of named-entities annotated with types is a task that received continuous attention over the years (Nothman et al., 2008; Al-Rfou et al., 2015; Ghaddar and Langlais, 2017). It consists mainly in exploiting the hyperlink structure of Wikipedia in order to detect entity mentions. Then, structured data from a knowledge base (for instance Freebase) are used to map hyperlinks to entity types. Because the number of anchored strings in Wikipedia is no more than 3% of the text tokens, Ghaddar and Langlais (2017) proposed to augment Wikipedia articles with mentions unmarked in Wikipedia, thanks to a mix of heuristics that benefit the Wikipedia structure (Ghaddar and Langlais, 2016a), as well as a coreference resolution system adapted specifically to Wikipedia (Ghaddar and Langlais, 2016b).

The authors applied their approach on English Wikipedia and produce coarse (4 classes) and fine-grained (120 labels) named-entity annotations, leading to WiNER (Ghaddar and Langlais, 2017) and WiFiNE (Ghaddar and Langlais, 2018). In this work, we adopt WiFiNE which is publicly available at `http://rali.iro.umontreal.ca/rali/en/wifiner-wikipedia-for-et` as our source of annotations. Each entity mention is mapped (via its Freebase `object_type` attribute) to a pre-defined set of 120 entity types. Types are stored in a 2-level hierarchical structure (e.g. `/person` and `/person/musician`). The corpus consist of 3.2M Wikipedia articles, comprising 1.3G tokens that we annotated with 157.4M named-entity mentions and their types. We used this very large quantity of automatically annotated data for jointly embedding words and entity types into the same low-dimensional space. The key idea consists in learning an embedding for each entity type using its surrounding words. For instance, the embedding for `/product/software` will be trained using context words that surround all entities that were (automatically) labelled as `/product/software` in Wikipedia. In practice, we found that simply concatenating a sentence (v1) with its annotated version (v2), as illustrated in Figure 1, offers a simple but efficient way of combining words and entity types so that embeddings can make good use of them.

(v1) On **October 9, 2009**, the **Norwegian Nobel Committee** announced that **Obama** had won the **2009 Nobel Peace Prize**.
(v2) On `/date`, the `/organization/government_agency` announced that `/person/politician` had won the `/award`.

Figure 1: Example of the two variants of a given sentence.

We use the FastText toolkit (Bojanowski et al., 2016) to learn the uncased embeddings for both words and entity types. We train a skipgram model to learn 100-dimensional vectors with a minimum word frequency cutoff of 5, and a window size of 5. This configuration (recommended by the authors) performs the best in the experiments described in Section 5. Since FastText learns representations of character *n*-grams, it has the ability to produce vectors for unknown words.

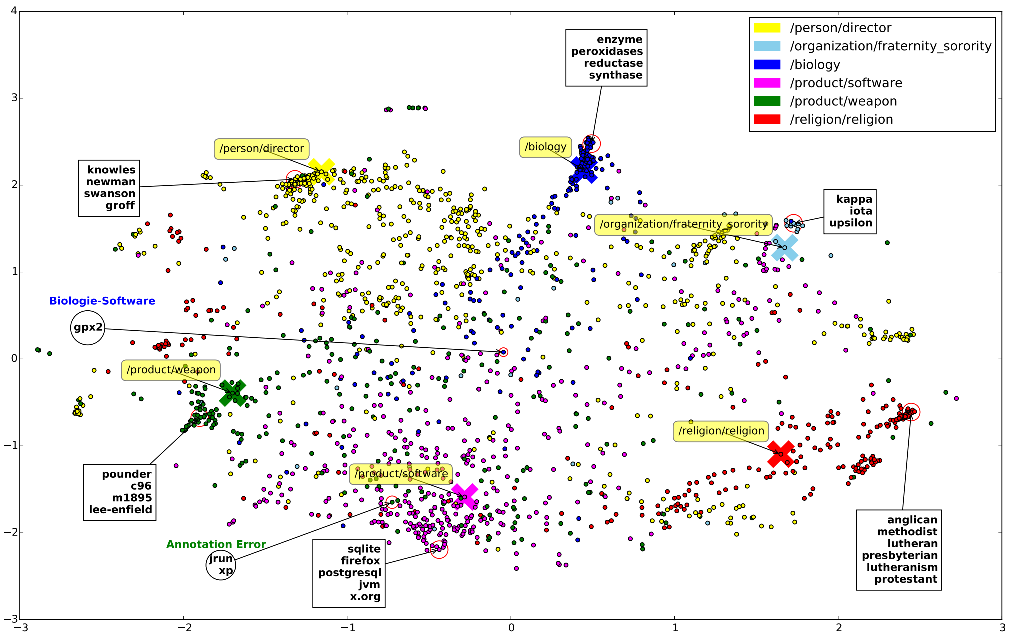

Figure 2: Two-dimensional representation of the vector space which embeds both words and entity types. Big Xs indicate entity types, while circles refer to words (i.e. named-entities, here).

Figure 2 illustrates a T-SNE (van der Maaten, 2014) two-dimensional projection of the embedding of 6 entity types and a sample of 1500 words. Entity type embeddings are marked by big Xs, while circles indicate words. For visualization proposes, we only plot single-word mentions that were annotated in WiFiNE with one of those 6 types. Words were randomly and proportionally sampled according to the frequency of each entity type. In addition, words have the color associated with the most frequent type they were annotated with in WiFiNE.

We observe that mentions often annotated by a given type in our resource tend to cluster around this entity type. For instance, "firefox" is close to the type `/product/software`, while "enzyme" is close to the `/biology` entity type. We also notice that words that are labelled with different types tend to appear between types they were annotated with. For instance, "gpx2", which is used both as a software and as a gene, has its embedding in between `/product/software` and `/biology`. We inspected some of the words plotted in Figure 2, and found that "jrun" and "xp" are incorrectly labelled as `/product/weapon` in WiFiNE. But since these words are seen in a *software* context, their embeddings are closer to the `/product/software` embedding than the `/product/weapon` one. We feel this tolerance to noise is a desirable feature, one that hopefully allows a more efficient use of distant supervision. Last, we also observe the tendency of rare words to cluster around their entity type. For instance, "iota" and "x.org" are embedded near their respective types, despite the fact that they appear less than 30 times in the version of Wikipedia used to compile WiFiNE.

3.2 LS Representation

This joint vector space only serves the purpose of associating to each word a LS representation, that is, a 120-dimensional vector where the ith coefficient is a value in the $[-1, +1]$ interval, equal to the cosine similarity[1] between the word embedding and the embedding of the ith entity type (we have 120 types).

Word	Entity Type	Sim	Word	Entity Type	Sim
hilton	`/building/hotel`	0.58	located	`/location`	0.47
	`/building/restaurant`	0.46		`/location/city`	0.44
	`/person/actor`	0.37		`/building`	0.40
gpx2	`/biology`	0.69	directed	`/person/director`	0.60
	`/product/software`	0.56		`/art/film`	0.55
jrun	`/product/software`	0.64	in	`/date`	0.58
	`/product/weapon`	0.23		`/location/city`	0.54
dammstadt	`/location/city`	0.45	won	`/award`	0.53
	`/location/railway`	0.44		`/event/sports_event`	0.53

Table 1: Topmost similar entity types to a few single-word mentions (left table) and non-entity words (right table).

Table 1 shows the topmost similar entity types for proper names (left column) and common words (right column). We observe that ambiguous mentions (those annotated with several types) are adequately handled. For instance, the LS representation of the word "hilton" encodes that it more often refers to a hotel or a restaurant than to an actress. Also, we observe that entity words that are either not or rarely annotated in WiFiNE are still adequately associated with their right type. For instance, "dammstadt", which appears only 5 times in WiFiNE, and which refers to the Damm city in Germany, is most similar to `/location/city` and `/location/railway`. Interestingly, this mention does not have its page in English Wikipedia. Furthermore, we observe that non-entity context words have a strong similarity to types they precede or succeed. For instance the verb "directed" is very close to `/person/director`, an entity type that usually precedes it, and to `/art/film`, that usually follows it. Likewise, the preposition "in" is near `/date` and `/location/city`, which frequently follow "in".

3.3 Strength of the LS Representation

To summarize, we propose a compact lexical representation which is computed offline, therefore incurring no computation burden at test time This representation encodes the preference of an entity-mention word for a given type, an information out of reach of binary gazetteer features. It also lends itself nicely to the inclusion of lexical features that have been successfully used in earlier feature-based systems (Ratinov and Roth, 2009; Luo et al., 2015). Also, because entity types are well represented in WiFiNE, their

[1]The cosine similarity outperforms other metrics in our experiments.

embeddings are robust: Our representation does accommodate unfrequent words and seems tolerant to the inherent noise of distant supervision.

4 Our NER System

In order to test the efficiency of our lexical feature representation, we implemented a state-of-the-art NER system we now describe.

4.1 Bi-LSTM-CRF Model

We adopt the popular Bi-LSTM-CRF architecture (Figure 3), a *de facto* baseline in many sequential tagging tasks (Lample et al., 2016; Søgaard and Goldberg, 2016; Chiu and Nichols, 2016).

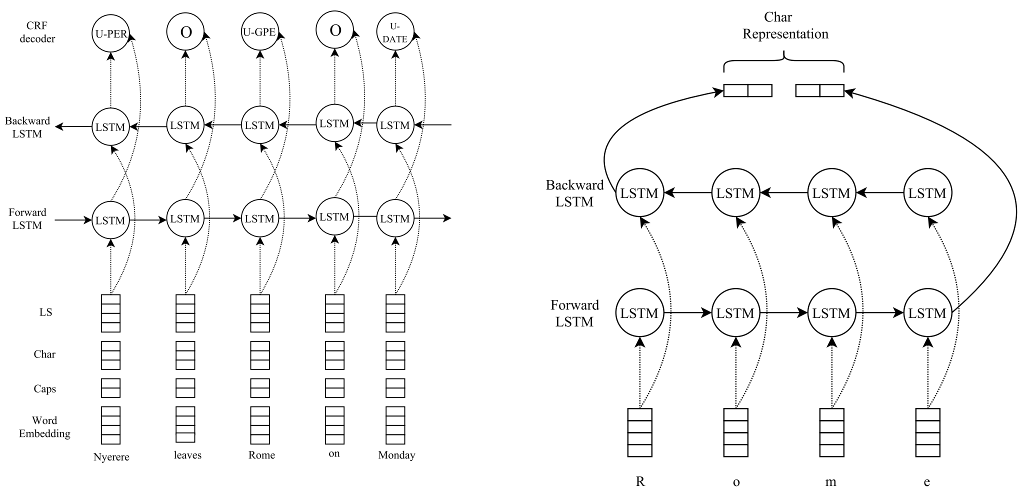

Figure 3: **Left Figure:** Main architecture of our NER system. **Right Figure:** Character representation of the word "Roma" given to the word-level bi-LSTM.

4.2 Features

In addition to the LS vector, we incorporate publicly available pre-trained embeddings, as well as character-level, and capitalization features. Those features have been shown to be crucial for state-of-the-art performance.

4.2.1 Word Embeddings

We experimented with several publicly available word embeddings, such as Senna (Collobert et al., 2011), Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and SSKIP (Yulia et al., 2015). We find that the latter performs the best in our experiments. SSKIP embeddings are 100-dimensional case sensitive vectors that where trained using a *n*-skip-gram model (Yulia et al., 2015) on 42B tokens. These embeddings were previously used by (Lample et al., 2016; Strubell et al., 2017), who report good performance on CONLL, and state-of-the-art results on ONTONOTES respectively. Note that these pre-trained embeddings are adjusted during training.

4.2.2 Character Embeddings

Following (Lample et al., 2016), we use a forward and a backward LSTM to derive a representation of each word from its characters (right part of Figure 3). A character lookup table is randomly initialized, then trained at the same time as the Bi-LSTM model sketched in Section 4.1.

4.2.3 Capitalization Features

Similarly to previous works, we use capitalization features for characterizing certain categories of capitalization patterns: `allUpper`, `allLower`, `upperFirst`, `upperNotFirst` , `numeric` or noAlphaNum. We define a random lookup table for these features, and learn its parameters during training.

4.2.4 LS Vectors

Contrarily to previous features, lexical vectors are computed offline and are not adjusted during training. We found useful in practice to apply a MinMax scaler in the range $[-1,+1]$ to each LS vector we computed; thus, $[..,0.095,..,0.20,..,0.76,..]$ becomes $[..,-1,..,-0.67,..,1,..]$.

5 Experiments

5.1 Data and Evaluation

We consider two well-established NER benchmarks: CoNLL-2003 and OntoNotes 5.0. Table 2 provides an overview of the two datasets. As we can see, OntoNotes is much larger. For both datasets, we convert the IOB encoding to BILOU, since Ratinov and Roth (2009) found the latter to perform better. In keeping with others, we report mention-level F1 score using the conlleval script[2].

The CoNLL-2003 NER dataset (Tjong Kim Sang and De Meulder, 2003) is a well known collection of Reuters newswire articles that contains a large portion of sports news. It is annotated with four entity types: *Person (*PER*), Location (*LOC*), Organization (*ORG*) and Miscellaneous (*MISC*)*. The four entity types are fairly evenly distributed, and the train/dev/test datasets present a similar type distribution.

Dataset		Train	Dev	Test
CoNLL-2003	*#tok*	204,567	51,578	46,666
	#ent	23,499	5,942	5,648
OntoNotes 5.0	*#tok*	1,088,503	147,724	152,728
	#ent	81,828	11,066	11,257

Table 2: Statistics of the CoNLL-2003 and OntoNotes 5.0 datasets. *#tok* stands for the number of tokens, and *#ent* indicates the number of named-entities gold annotated.

The OntoNotes 5.0 dataset (Hovy et al., 2006; Pradhan et al., 2013) includes texts from five different genres: broadcast conversation (200k), broadcast news (200k), magazine (120k), newswire (625k), and web data (300k). This dataset is annotated with 18 entity types, and is much larger than CoNLL. Following previous researches (Chiu and Nichols, 2016; Strubell et al., 2017), we use the official train/dev/test split of the CoNLL-2012 shared task (Pradhan et al., 2012). Also, we exclude (both during training and testing) the New Testaments portion as it does not contain gold NE annotations.

5.2 Training and Implementation

Training is carried out by mini-batch stochastic gradient descent (SGD) with a momentum of 0.9 and a gradient clipping of 5.0. The mini-batch is 10 for both datasets, and learning rates are 0.009 and 0.013 for CoNLL and OntoNotes respectively. More sophisticated optimization algorithms such as AdaDelta (Zeiler, 2012) or Adam (Kingma and Ba, 2014) converge faster, but none outperformed SGD with exponential learning rate decay in our experiments.

Our system uses a single Bi-LSTM layer at the word level whose hidden dimensions are set to 128 and 256 for CoNLL and OntoNotes respectively. For both models, the character embedding size was set to 25, and the hidden dimension of the forward and backward character LSTMs are set to 50. To mitigate overfitting, we apply a dropout mask (Srivastava et al., 2014) with a probability of 0.5 on the input and output vectors of the Bi-LSTM layer. For both datasets, we set the dimension of capitalization embeddings to 25 and trained the models up to 50 epochs.

We tuned the hyper-parameters by grid search, and used early stopping based on the performance on the development set. We varied dropout ($[0.25, 0.5, 0.65]$), hidden units ($[50, 128, 256, 300]$), capitalization ($[10, 20, 30]$) and char ($[25, 50, 100]$) embedding dimensions, learning rate ($[0.001, 0.015]$ by step

[2]http://www.cnts.ua.ac.be/conll2000/chunking/conlleval.txt

0.002), and optimization algorithms and fixed the other hyper-parameters. We implemented our system using the Tensorflow (Abadi et al., 2016) library, and ran our models on a GeForce GTX TITAN Xp GPU. Training requires about 2.5 hours for CoNLL and 8 hours for OntoNotes.

5.3 Results on the Development Set

Table 3 shows the development set performance of our final models on each dataset compared to the work of Chiu and Nichols (2016). The authors use an architecture similar to ours, but use a binary gazetteer feature set, while we use our LS representation. Since our systems involve random initialization, we report the mean as well as the standard deviation over five runs. The improvements yielded by our model on the CoNLL dataset are significant although modest, while those observed on OntoNotes are more substantial. We also observe a lower variance of our system over the 5 runs.

	CoNLL	**OntoNotes**
(Chiu and Nichols, 2016)	94.03 (± 0.23)	84.57 (± 0.27)
Our model	**94.80 (± 0.10)**	**86.44 (± 0.14)**

Table 3: Development set F1 scores of our best hyper-parameter setting compared to the results reported in (Chiu and Nichols, 2016).

5.4 Results on CoNLL

Table 4 reports our model's performance[3]on the CoNLL test set, as well as the performance of systems previously tested on this test set (the figures are those published by the authors). Because of the small size of the training set, some authors (Chiu and Nichols, 2016; Yang et al., 2017; Peters et al., 2017; Peters et al., 2018) incorporated the development set as a part of training data after tuning the hyper-parameters. Consequently, their results are not directly comparable, so we do not report them.

Model	LEX	GAZ	CAP	EMB	CHE	LME	LS	**F1**
(Finkel et al., 2005)	+	+	+	●	●	●	●	86.86
(Ratinov and Roth, 2009)	+	+	+	●	●	●	●	90.88
(Lin and Wu, 2009)	+	+	+	●	●	●	●	90.90
(Luo et al., 2015)	+	+	+	●	●	●	●	91.20
(Collobert et al., 2011)	●	+	+	+	●	●	●	89.56
(Huang et al., 2015)	●	●	+	+	+	●	●	90.10
(Lample et al., 2016)	●	●	+	+	+	●	●	90.94
(Ma and Hovy, 2016)	●	●	+	+	+	●	●	91.21
(Shen et al., 2017)	●	●	+	+	●	●	●	90.89
(Strubell et al., 2017)	●	●	+	+	●	●	●	90.54
(Tran et al., 2017)	●	●	+	+	+	+	●	91.69
(Liu et al., 2017)	●	●	+	+	+	+	●	91.71
This work	●	+	+	+	+	●	+	**91.73**

Table 4: F1 scores on the CoNLL test set. The first four systems are feature-based, the others are neuronal. The feature configuration of each system is encoded with: LEX which stands for LEXical feature, GAZ for GAZetteers, CAP for CAPitalization, EMB for pre-trained EMBeddings, CHE for CHaracter Embeddings, LME for Language Model Embeddings, and LS for the proposed LS feature representation. + indicates that the model uses the feature set.

First, we observe that our model significantly outperforms models that use extensive sets of hand-crafted features (Ratinov and Roth, 2009; Lin and Wu, 2009) as well as the system of (Luo et al.,

[3]Standard deviation on the test set is reported in Table 7

2015) that uses NE and Entity Linking annotations to jointly optimize the performance on both tasks. Second, our model outperforms as well other NN models that only use standard word embeddings, which indicates that our lexical feature vector is complementary to standard word embeddings. Third, our system matches state-of-the-art performances of models that use either more complex architectures or more elaborate features. Tran et al. (2017) use three layers of stacked residual RNN (Bi-LSTM) with bias decoding. Our model is much simpler and faster. They report a performance of 90.43 when using an architecture similar to ours. The two systems that have slightly higher F1 scores on the CoNLL dataset both use embeddings obtained from a forward and a backward Language Model trained on the One Billion Word Benchmark (Chelba et al., 2013). They report gains between 0.8 and 1.2 points by using such LM embeddings, which suggests that LS vectors are indeed efficient. Unfortunately, due to time and resource constraints,[4] we were not able to measure whether both features complement each other. This is left for future investigations.

5.5 Results on ONTONOTES

Table 5 reports the F1 score of our system compared to the performance reported by others on the ONTONOTES test set. To the best of our knowledge, we surpass previously reported F1 scores on this dataset. In particular, our system significantly outperforms the Bi-LSTM-CNN-CRF models of (Chiu and Nichols, 2016) and (Strubell et al., 2017) by an absolute gain of 1.68 and 0.96 points respectively. Less surprisingly, it surpasses systems with hand-crafted features, including Ratinov and Roth (2009) that use gazetteers, and the system of Durrett and Klein (2014) which uses coreference annotation in ONTONOTES to jointly model NER, entity linking, and coreference resolution tasks.

Model	LEX	GAZ	CAP	EMB	CHE	LME	LS	**F1**
(Finkel and Manning, 2009)	+	+	+	●	●	●	●	82.42
(Ratinov and Roth, 2009)	+	+	+	●	●	●	●	84.88
(Passos et al., 2014)	+	+	+	●	●	●	●	82.24
(Durrett and Klein, 2014)	+	+	+	●	●	●	●	84.04
(Chiu and Nichols, 2016)	●	+	+	+	+	●	●	86.28
(Shen et al., 2017)	●	●	+	+	+	●	●	86.52
(Strubell et al., 2017)	●	●	+	+	+	●	●	86.99
This work	●	+	+	+	+	●	+	**87.95**

Table 5: F1 scores on the ONTONOTES test set. The first four systems are feature-based, the following ones are neuronal. See Table 4 for an explanation of the column of features.

The ONTONOTES benchmark is annotated with 18 types (e.g. LAW, PRODUCT) and contains many rare words, especially in the Web data collection. Chiu and Nichols (2016) note that the 4-class gazetteer they used yielded marginal improvements on ONTONOTES, contrarily to CoNLL. In particular, they observe that mentions that match LOC entries in their gazetteer often match GPE, NORP and FAC lists.

Model	**BC**	**BN**	**MZ**	**NW**	**TC**	**WB**
(Finkel and Manning, 2009)	78.66	87.29	82.45	85.50	67.27	72.56
(Durrett and Klein, 2014)	78.88	87.39	82.46	87.60	72.68	76.17
(Chiu and Nichols, 2016)	85.23	89.93	84.45	88.39	72.39	78.38
This work	**86.33**	**90.46**	**85.91**	**89.75**	**75.41**	**80.39**

Table 6: Per-genre F1 scores on ONTONOTES (numbers taken from Chiu and Nichols (2016)). BC = broadcast conversation, BN = broadcast news, MZ = magazine, NW = newswire, TC = telephone conversation, WB = blogs and newsgroups.

[4]LM embeddings are not publicly available, and according to Jozefowicz et al. (2016), they require three weeks to train on 32 GPUs.

They suggest that a finer-grained gazetteer could improve the performance of their system on ONTONOTES. Our results confirm this, since we use 120 types. We further detail the gains we observed for each sub-collection of texts in the test set. Table 6 reveals that major improvements over the model of (Chiu and Nichols, 2016) are on noisier collections such as telephone conversations (+3 points) and blogs or newsgroups (+2 points). Those type of texts are characterized by a large set of infrequent words, for which classical embeddings are typically poorly trained. Our approach does not seem to suffer from this problem as severely, as discussed in Section 2.

5.6 Ablation Results

In this experiment, we directly compare the LS representation with the SSKIP word-embedding feature set. In order to maintain a high level of performance, both character and capitalization features are used in all configurations. We want to point out that LS vectors are not adapted during training, contrarily to the SSKIP embeddings. Similarly to Section 5.3, we report in Table 7, for each feature configuration, the average F1 score as well as the standard deviation over five runs.

Model	**CONLL**	**ONTONOTES**
SSKIP	90.52 (± 0.18)	86.57 (± 0.10)
LS	89.94 (± 0.16)	85.92 (± 0.12)
all	**91.73 (± 0.10)**	**87.95 (± 0.13)**

Table 7: F1 scores of differently trained systems on CONLL and ONTONOTES 5.0 datasets. Capitalization (Section 4.2.3) and character features (Section 4.2.2) are used by default by all models.

We observe that on both CONLL and ONTONOTES, the SSKIP model outperforms our feature vector approach by 0.65 F1 points on average. The difference is not has high as we first expected, especially since the SSKIP model is adjusted during training, while our representation is not. Still, LS vectors seem to encode a large portion of the information needed to model the NER task. Also, it is worth mentioning that our embeddings are trained on 1.3B words compared to 42B for SSKIP.

We also observe that models that use both feature sets significantly outperform other configurations. To confirm that the gains came from our feature vector and not from increasing the number of hidden units, we tested several SSKIP models by increasing the LSTM hidden layer dimension so that number of parameters is the same as the model with LS vectors. We observed a degradation of performance on both datasets, mostly due to overfitting on the training set. From those results, we conclude that our lexical representation and the SSKIP one are complementary.

6 Related Works

Traditional approaches to NER, like CRF-based (Finkel et al., 2005) and Perceptron-based systems (Ratinov and Roth, 2009) have dominated the field for over a decade. They rely heavily on hand-engineered features (Luo et al., 2015) and external resources such as gazetteers. One major drawback of such an approach is its weak generalization power (Lample et al., 2016). Current state-of-the art systems (Chiu and Nichols, 2016; Strubell et al., 2017) use a combination of Convolutional Neural Networks (CNNs), Bi-LSTMs, along with a CRF decoder. CNNs are used to encode character-level features (prefix and suffix), while LSTM is used to encode word-level features. Finally, a CRF is placed on top of those models in order to decode the best tag sequence. Pre-trained embeddings obtained by unsupervised learning are core features of those models. In this work, we show that deep NN architectures can also benefit from lexical features, at least when encoded in the compact form we propose.

Tran et al. (2017) and Peters et al. (2017) propose an alternative approach different from ours. They incorporate LM embeddings that were pre-trained on a large unlabelled corpus as features for NER. These embeddings allow to generate a representation for a word depending on its context. For instance, the LM embeddings of the word *France* in "*France is a developed country*" is different than that in "*Anatole France began his literary career*". Such embeddings are trained on very large amount of texts.

Our feature set is crafted from distant supervision applied to Wikipedia, a much less time-consuming process which we showed to be nevertheless adapted to rare words. Chiu and Nichols (2016) used gazetteer features in order to establish state-of-the-art performance on both CoNLL and OntoNotes. They mined DBPedia in order to compile 4 lists of named-entities that contain over 2.3M entries. We show that LS representations outperform their gazetteer features.

7 Conclusion and Future Work

We have explored the idea of generating lexical features for NER out of Wikipedia data automatically annotated with fine-grained entity types. We used WiFiNE (Ghaddar and Langlais, 2018), a Wikipedia dump annotated with fine entity type mentions, for training a vector space that jointly embeds words and named-entities. This vector space is used to compute a 120 dimensional vector per word, which encodes the similarity of the word to each of the entity types. Our results show that our proposed lexical representation, even though it is not adjusted at training time, matches state-of-the-art results compared to more complex approaches on the well-studied CoNLL dataset, and delivers a new state-of-the-art F1 score of 87.95 on the more diversified OntoNotes dataset. We further observe larger gains on collections with more unfrequent words.

The source code and the data we used in this work are publicly available at `http://rali.iro.umontreal.ca/rali/en/wikipedia-lex-sim`, with the hope that other researchers will report gains, when using our lexical representation. As a future work, we want to investigate the usefulness of our LS feature representation on other NER tasks, including NER in tweets where out-of-vocabulary and low-frequency words represent a challenge; as well as finer-grained NER which suffers from the lack of manually annotated training data.

Acknowledgements

This work has been partly funded by the TRIBE Natural Sciences and Engineering Research Council of Canada CREATE program and Nuance Foundation. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research. We thank the anonymous reviewers for their insightful comments.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Rami Al-Rfou, Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. 2015. Polyglot-NER: Massive multilingual named entity recognition. In *Proceedings of the 2015 SIAM International Conference on Data Mining, Vancouver, British Columbia, Canada*. SIAM.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. In *Proceedings of the 54st Annual Meeting of the Association for Computational Linguistics*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Greg Durrett and Dan Klein. 2014. A Joint Model for Entity Analysis: Coreference, Typing, and Linking. *Transactions of the Association for Computational Linguistics*, 2:477–490.

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.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Abbas Ghaddar and Philippe Langlais. 2016a. Coreference in Wikipedia: Main Concept Resolution. In *CoNLL*, pages 229–238.

Abbas Ghaddar and Philippe Langlais. 2016b. WikiCoref: An English Coreference-annotated Corpus of Wikipedia Articles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia, 05/2016.

Abbas Ghaddar and Phillippe Langlais. 2017. WiNER: A Wikipedia Annotated Corpus for Named Entity Recognition. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 413–422.

Abbas Ghaddar and Phillippe Langlais. 2018. Transforming Wikipedia into a Large-Scale Fine-Grained Entity Type Corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France, may. European Language Resources Association (ELRA), European Language Resources Association (ELRA).

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60. Association for Computational Linguistics.

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

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for Named Entity Recognition. *arXiv preprint arXiv:1603.01360*.

Dekang Lin and Xiaoyun Wu. 2009. Phrase Clustering for Discriminative Learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1030–1038. Association for Computational Linguistics.

Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2017. Empower Sequence Labeling with Task-Aware Neural Language Model. *arXiv preprint arXiv:1709.04109*.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint Entity Recognition and Disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 879–888, Lisbon, Portugal, September. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Joel Nothman, James R Curran, and Tara Murphy. 2008. Transforming Wikipedia into named entity training data. In *Proceedings of the Australian Language Technology Workshop*, pages 124–132.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for Named Entity Resolution. *arXiv preprint arXiv:1404.5367*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*, volume 14, pages 1532–1543.

Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards Robust Linguistic Analysis using OntoNotes. In *CoNLL*, pages 143–152.

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.

Tom Redman, Mark Sammons, and Dan Roth. 2016. Illinois Named Entity Recognizer: Addendum to Ratinov and Roth '09 reporting improved results.

Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep Active Learning for Named Entity Recognition. *arXiv preprint arXiv:1707.05928*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 231–235.

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.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and Accurate Entity Recognition with Iterated Dilated Convolutions. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2660–2670.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Quan Tran, Andrew MacKinlay, and Antonio Jimeno Yepes. 2017. Named entity recognition with stack residual lstm and trainable bias decoding. *arXiv preprint arXiv:1706.07598*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

L.J.P. van der Maaten. 2014. Accelerating t-SNE using Tree-Based Algorithms. *Journal of Machine Learning Research*, 15:3221–3245.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural Reranking for Named Entity Recognition. *arXiv preprint arXiv:1707.05127*.

Wang Ling Lin Chu-Cheng Yulia, Tsvetkov Silvio Amir, Ramón Fernandez Astudillo Chris Dyer Alan, and W Black Isabel Trancoso. 2015. Not all contexts are created equal: Better word representations with variable attention.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.