Multimedia Lab @ ACL W-NUT NER Shared Task: Named Entity Recognition for Twitter Microposts using Distributed Word Representations

Fréderic Godin
Multimedia Lab
Ghent University - iMinds
Ghent, Belgium
frederic.godin@ugent.be

Baptist Vandersmissen
Multimedia Lab
Ghent University - iMinds
Ghent, Belgium
baptist.vandersmissen@ugent.be

Wesley De Neve
Multimedia Lab & IVY Lab
Ghent University - iMinds & KAIST
Ghent, Belgium & Daejeon, South-Korea
wesley.deneve@ugent.be

Rik Van de Walle
Multimedia Lab
Ghent University - iMinds
Ghent, Belgium
rik.vandewalle@ugent.be

Abstract

Due to the short and noisy nature of Twitter microposts, detecting named entities is often a cumbersome task. As part of the ACL2015 Named Entity Recognition (NER) shared task, we present a semi-supervised system that detects 10 types of named entities. To that end, we leverage 400 million Twitter microposts to generate powerful word embeddings as input features and use a neural network to execute the classification. To further boost the performance, we employ dropout to train the network and leaky Rectified Linear Units (ReLUs). Our system achieved the fourth position in the final ranking, without using any kind of hand-crafted features such as lexical features or gazetteers.

1 Introduction

Users on Online Social Networks such as Facebook and Twitter have the ability to share microposts with their friends or followers. These microposts are short and noisy, and are therefore much more difficult to process for existing Natural Language Processing (NLP) pipelines. Moreover, due to the informal and contemporary nature of these microposts, they often contain Named Entities (NEs) that are not part of any gazetteer.

In this challenge, we tackled Named Entity Recognition (NER) in microposts. The goal was to detect named entities and classify them in one of the following 10 categories: company, facility, geolocation, music artist, movie, person, product, sports team, tv show and other entities. To do so, we *only* used word embeddings that were automatically inferred from 400 million Twitter microposts as input features. Next, these word embeddings were used as input to a neural network to classify the words in the microposts. Finally, a post-processing step was executed to check for inconsistencies, given that we classified on a word-per-word basis and that a named entity can span multiple words. An overview of the task can be found in Baldwin et al. (2015).

The challenge consisted of two subtasks. For the first subtask, the participants only needed to detect NEs without categorizing them. For the second subtask, the NEs also needed to be categorized into one of the 10 categories listed above. Throughout the remainder of this paper, only the latter subtask will be considered, given that solving subtask two makes subtask one trivial.

2 Related Work

NER in news articles gained substantial popularity with the CoNLL 2003 shared task, where the challenge was to classify four types of NEs: persons, locations, companies and a set of miscellaneous entities (Tjong Kim Sang and De Meulder, 2003). However, all systems used hand-crafted features such as lexical features, look-up tables and corpus-related features. These systems provide good performance at a high engineering cost and need a lot of annotated training data (Nadeau and Sekine, 2007). Therefore, a lot of effort is needed to adapt them to other types of corpora.

More recently, semi-supervised systems

Proceedings of the ACL 2015 Workshop on Noisy User-generated Text, pages 146–153,
Beijing, China, July 31, 2015.

showed to achieve near state-of-the-art results with much less effort (Turian et al., 2010; Collobert et al., 2011). These systems first learn word representations from large corpera in an unsupervised way and use these word representations as input features for supervised training instead of using hand-crafted input features. There exist three major types of word representations: distributional, clustering-based and distributed word representations, and where the last type of representation is also known as a word embedding. A very popular and fast to train word embedding is the word2vec word representation of Mikolov et al. (2013). When complemented with traditional hand-crafted features, word representations can yield F1-scores of up to 91% (Tkachenko and Simanovsky, 2012).

However, when applied to Twitter microposts, the F1-score drops significantly. For example, Liu et al. (2011) report a F1 score of 45.8% when applying the Stanford NER tagger to Twitter microposts and Ritter et al. (2011) even report a F1-score of 29% on their Twitter micropost dataset. Therefore, many researchers (Cano et al., 2013; Cano et al., 2014) trained new systems on Twitter microposts, but mainly relied on cost-intensive handcrafted features, sometimes complemented with cluster-based features.

Therefore, in this paper, we will investigate the power of word embeddings for NER applied to microposts. Although adding hand-crafted features such as lexical features or gazetteers would probably improve our F1-score, we will only focus on word embeddings, given that this approach can be easily applied to different corpora, thus quickly leading to good results.

3 System Overview

The system proposed for tackling this challenge consists of three steps. First, the individual words are converted into word representations. For this, only the word embeddings of Mikolov et al. (2013) are used. Next, we feed the word representations to a Feed-Forward Neural Network (FFNN) to classify the individual words with a matching tag. Finally, we execute a simple, rule-based post-processing step in which we check the coherence of individual tags within a Named Entity (NE).

3.1 Creating Feature Representations

Recently, Mikolov et al. (2013) introduced an efficient way for inferring word embeddings that are effective in capturing syntactic and semantic relationships in natural language. In general, a word embedding of a particular word is inferred by using the previous or future words within a number of microposts/sentences. Mikolov et al. (2013) proposed two architectures for doing this: the Continuous Bag Of Words (CBOW) model and the Skip-gram model.

To infer the word embeddings, a large dataset of microposts is used. The algorithm iterates a number of times over this dataset while updating the word embeddings of the words within the vocabulary of the dataset. The final result is a look-up table which can be used to convert every word $w(t)$ in a feature vector $w_e(t)$. If the word is not in the vocabulary, a vector only containing zeros is used.

3.2 Neural Network Architecture

Based on the successful application of Feed-Forward Neural Networks (FFNN) using word embeddings as input features for both recognizing NEs in news articles (Turian et al., 2010; Collobert et al., 2011) and Part-of-Speech tagging of Twitter microposts (Godin et al., 2014), a FFNN is used as the underlying classification algorithm. Because a NE can consist of multiple words, the BIO (Begin, Inside, Outside NE) notation is used to classify the words. Given that there are 10 different NE categories that each have a Begin and Inside tag, the FFNN will assign a $tag(t)$ to every word $w(t)$ out of 21 different tags.

Because the tag $tag(t)$ of a word $w(t)$ is also determined by the surrounding words, a context window centered around $w(t)$ that contains an odd number of words, is used. As shown in Figure 1, the corresponding word embeddings $w_e(t)$ are concatenated and used as input to the FFNN. This is the input layer of the neural network.

The main design parameters of the neural network are the type of activation function, the number of hidden units and the number of hidden layers. We considered two types of activation functions, the classic *tanh* activation function and the newer (leaky) Rectified Linear Units (ReLUs). The output layer is a standard softmax function.

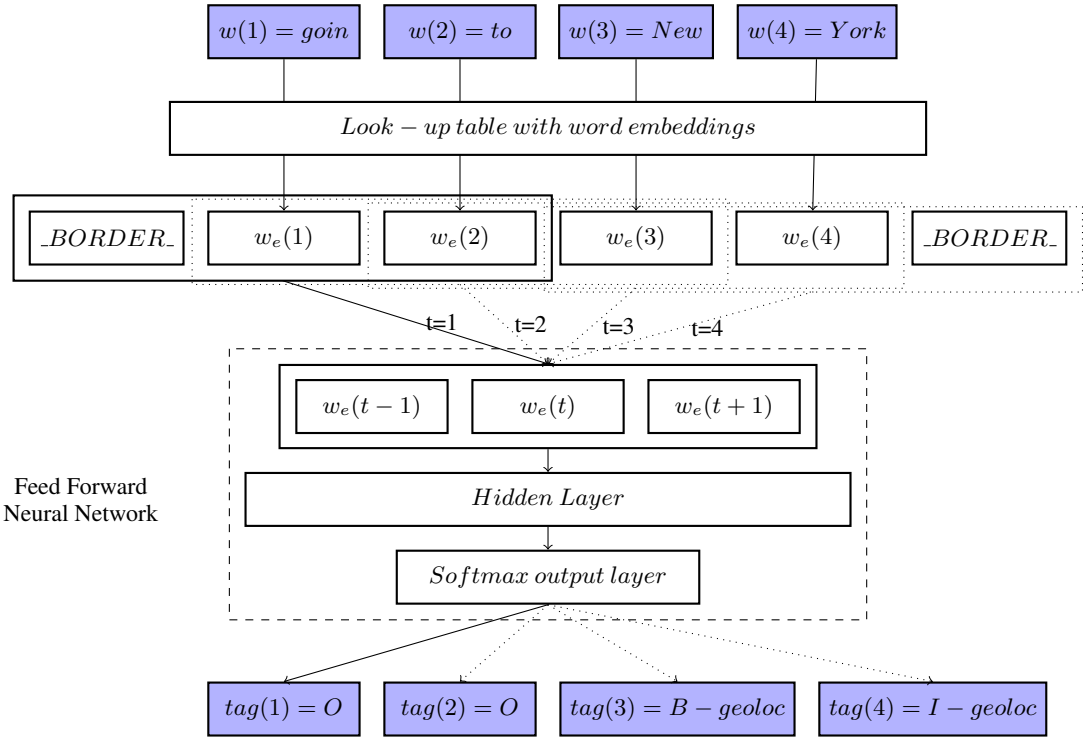

Figure 1: High-level illustration of the FFNN that classifies each word as part of one of the 10 named entity classes. At the input, a micropost containing four words is given. The different words $w(t)$ are first converted in feature representations $w_e(t)$ using a look-up table of word embeddings. Next, a feature vector is constructed for each word by concatenating all the feature representations $w_e(t)$ of the other words within the context window. In this example, a context window of size three is used. One-by-one, these concatenated vectors are fed to the FFNN. In this example, a one-hidden layer FFNN is used. The output of the FFNN is the tag $tag(t)$ of the corresponding word $w(t)$.

3.3 Postprocessing the Neural Network Output

Given that NEs can span multiple words and given that the FFNN classifies individual words, we apply a postprocessing step after a micropost is completely classified to correct inconsistencies. The tags of the words are changed according to the following two rules:

- If the NE does not start with a word that has a B(egin)-tag, we select the word before the word with the I(nside)-tag and replace the O(utside)-tag with a B-tag and copy the category of the I-tag.

- If the individual words of a NE have different categories, we select the most frequently occurring category. If it is a tie, we select the category of the last word within the NE.

4 Experimental Setup

4.1 Dataset

The challenge provided us with three different datasets: train, dev and dev_2015. These datasets have 1795, 599 and 420 microposts, respectively, also containing 1140, 356 and 272 NEs, respectively. The train and dev datasets came from the same period and therefore have some overlap in NEs. Moreover, they contained the complete dataset of Ritter et al. (2011). The microposts within dev_2015, however, were sampled more recently and resembled the test set of this challenge. The test set consisted of 1000 microposts, having 661 NEs. The train and dev dataset will be used as training set throughout the experiments and the dev_2015 dataset will be used as development set.

For inferring the word embeddings, a set of raw Twitter microposts was used, collected during 300 days using the Twitter Streaming API, from

1/3/2013 till 28/2/2014. After removing all non-English microposts using the micropost language classifier of Godin et al. (2013), 400 million raw English Twitter microposts were left.

4.2 Preprocessing the Data

For preprocessing the 400 million microposts, we used the same tokenizer as Ritter et al. (2011). Additionally, we used replacement tokens for URLs, mentions and numbers on both the challenge dataset and the 400 million microposts we collected. However, we did not replace hashtags as doing so experimentally demonstrated to decrease the accuracy.

4.3 Training the Model

The model was trained in two phases. First, the look-up table containing per-word feature vectors was constructed. To that end, we applied the word2vec software (v0.1c) of Mikolov et al. (2013) on our preprocessed dataset of 400 million Twitter microposts to generate word embeddings. Next, we trained the neural network. To that end, we used the Theano library (v0.6) (Bastien et al., 2012), which easily effectuated the use of our NVIDIA Titan Black GPU. We used mini-batch stochastic gradient descent with a batch size of 20, a learning rate of 0.01 and a momentum of 0.5. We used the standard negative log-likelihood cost function to update the weights. We used dropout on both the input and hidden layers to prevent overfitting and used (Leaky) Rectified Linear Units (ReLUs) as hidden units (Srivastava et al., 2014). To do so, we used the implementation of the Lasagne[1] library. We trained the neural network on both the train and dev dataset and iterated until the accuracy on the dev_2015 set did not improve anymore.

4.4 Baseline

To evaluate our system, we made use of two different baselines. The word embeddings and the neural network architecture were evaluated in terms of word level accuracy. For these components, the baseline system simply assigned the O-tag to every word, yielding an accuracy of 93.53%. For the postprocessing step and the overall system evaluation, we made use of the baseline provided by the challenge, which performs an evaluation at the level of NEs. This baseline system uses lexical features and gazetteers, yielding an F1-score of 34.29%. Note that we only report the performance for subtask two (i.e., categorizing the NEs), except for the final evaluation.

[1] https://github.com/Lasagne/Lasagne

Table 1: Evaluation of the influence of the context window size of the word embeddings on the accuracy of predicting NER tags using a neural network with an input window of five words, 500 hidden Leaky ReLU units and dropout. All word embeddings are inferred using negative sampling and a Skip-gram architecture, and have a vector size of 400. The baseline accuracy is achieved when tagging all words of a micropost with the O-tag.

Context Window	Accuracy	Error Rate Reduction
Baseline	93.53%	
1	**95.64%**	**-32.57%**
3	95.57%	-31.44%
5	95.52%	-30.72%

5 Experiments

5.1 Word Embeddings

For inferring the word embeddings of the 400 million microposts, we mainly followed the suggestions of Godin et al. (2014), namely, the best word embeddings are inferred using a Skip-gram architecture and negative sampling. We used the default parameter settings of the word2vec software, except for the context window.

As noted by Bansal et al. (2014), the type of word embedding created depends on the size of the context window. In particular, a bigger context window creates topic-oriented embeddings while a smaller context window creates syntax-oriented embeddings. Therefore, we trained an initial version of our neural network using an input window of five words and 300 hidden nodes, and evaluate the quality of the word embeddings based on the classification accuracy on the dev_2015 dataset. The results of this evaluation are shown in Table 1. Although the difference is small, a smaller context window consistently gave a better result.

Additionally, we evaluated the vector size. As a general rule of thumb, the larger the word embeddings, the better the classification (Mikolov et al., 2013). However, too many parameters and too few training examples will lead to suboptimal re-

sults and poor generalization. We chose word embeddings of size 400 because smaller embeddings experimentally showed to capture not as much detail and resulted in a lower accuracy. Larger word embeddings, on the other hand, made the model too complex to train.

The final word2vec word embeddings model has a vocabulary of 3,039,345 words and word representations of dimensionality 400. The model was trained using the Skip-gram architecture and negative sampling ($k = 5$) for five iterations, with a context window of one and subsampling with a factor of 0.001. Additionally, to be part of the vocabulary, words should occur at least five times in the corpus.

5.2 The Neural Network Architecture

The next step is to evaluate the Neural Network Architecture. The most important parameters are the size of the input layer, the size of the hidden layers and the type of hidden units. Although we experimented with multiple hidden layers, the accuracy did not improve. We surmise that (1) the training set is too small and that (2) the word embeddings already contain a fixed summary of the information and therefore limit the feature learning capabilities of the neural network. Note that the word embeddings at the input layer can also be seen as a (fixed) hidden layer.

First, we will evaluate the activation function and the effectiveness of dropout ($p = 0.5$). We compared the classic *tanh* function and the leaky ReLU with a leak rate of 0.01[2]. As can be seen in Table 2, both activation functions perform equally good when no dropout is applied during training. However, when dropout is applied, the gap between the two configurations becomes larger. The combination of ReLUs and dropout seems to be the best one, compared to the classic configuration of a neural network[3].

Next, we will evaluate a number of neural network configurations for which we varied the input layer size and the hidden layer size. The results are depicted in Table 3. Although the differences are small, the best configuration seems to be a neural network with five input words and 500 hidden nodes.

[2]This is the default value in Lasagne and showed to work the best for us

[3]Given that dropout also acts as a regularization technique, a comparison with other regularization techniques should be conducted to be complete (e.g., L2 regularization)

Table 2: Evaluation of the influence of the activation function and dropout on the accuracy of predicting NE tags. A fixed neural network with an input window of five, word embeddings of size 400 and 500 hidden units is used. The baseline accuracy is achieved when tagging all words of a micropost with the O-tag.

Activ. Function	Dropout	Accuracy	Error Rate Reduction
Baseline		93.53%	
Tanh	No	95.01%	-22.78%
	Yes	95.49%	-30.30%
L. ReLU	No	95.02%	-23.01%
	Yes	**95.64%**	**-32.57%**

Finally, we evaluate our best model on the NE level instead of on the word level. To that end, we calculated the F1-score of our best model using the provided evaluation script. The F1-score of our best model on dev_2015 is 45.15%, which is an absolute improvement of almost 11% and an error reduction of 16.52% over the baseline (34.29%) provided by the challenge.

5.3 Postprocessing

As a last step, we corrected the output of the neural network for inconsistencies because our classifier does not see the labels of the neighbouring words. As can be seen in Table 4, the postprocessing causes a significant error reduction, yielding a final F1-score of 49.09% on the dev_2015 development set, and an error reduction of 22.52% over the baseline.

Additionally, we also report the F1-score on the dev_2015 development set for subtask one, where the task was to only detect the NEs but not to categorize them into the 10 different categories. For this, we retrained the model with the best parameters of subtask two. The results are shown in Table 5.

If we compare both subtasks, we see that the neural network has a similar error reduction for both subtasks but that the postprocessing step effectuates a larger error reduction for subtask two. In other words, a common mistake of the neural network is to assign different categories to different words within one NE. These mistakes are easily corrected by the postprocessing step.

Table 3: Evaluation of the influence of the input layer and hidden layer size on the accuracy/error reduction when predicting NE tags. The fixed neural network is trained with dropout, word embeddings of size 400 and ReLUs. The error reduction values are calculated using the baseline which tags all words with an O-tag.

Window	Number of hidden units		
	300	500	1000
Three	95.63% / -32.35%	95.58% / -31.66%	95.55% / -31.21%
Five	95.57% / -31.44%	**95.64% / -32.57%**	95.61% / -32.12%

Table 4: Evaluation of the postprocessing step for detecting named entities. The baseline was provided by the challenge.

Configuration	F1	Error Rate Reduction
Baseline	34.29%	
Without postprocessing	45.15%	-16.52%
With postprocessing	**49.09%**	**-22.52%**

Table 5: Evaluation of the postprocessing step for detecting named entities without categorizing them. The baseline was provided by the challenge.

Configuration	F1	Error Rate Reduction
Baseline	52.63%	
Without postprocessing	60.04%	-15.64%
With postprocessing	**60.80%**	**-17.24%**

6 Evaluation on the Test Set

Our best model realized a F1-score of 58.82% on subtask one (no categories), hereby realizing an error reduction of 17.84% over the baseline (49.88%). On subtask two (10 categories), an F1-score of 43.75% was realized, yielding an error reduction of 17.32% over the baseline (31.97%). A break-down of the results on the different NE categories can be found in Table 6. Our system ranked fourth in both subtasks.

7 Conclusion

In this paper, we presented a system to apply Named Entity Recognition (NER) to microposts. Given that microposts are short and noisy compared to news articles, we did not want to invest time in crafting new features that would improve NER for microposts. Instead, we implemented the semi-supervised architecture of Collobert et al. (2011) for NER in news articles. This architecture only relies on good word embeddings inferred from a large corpus and a simple neural network.

To realize this system, we used the word2vec software to quickly generate powerful word embeddings over 400 million Twitter microposts. Additionally, we employed a state-of-the-art neural network for classification purposes, using leaky Rectified Linear Units (ReLUs) and dropout to train the network, showing a significant benefit over classic neural networks. Finally, we checked the output for inconsistencies when categorizing the named entities.

Our word2vec word embeddings trained on 400 million microposts are released to the community and can be downloaded at http://www.fredericgodin.com/software/.

Acknowledgments

The research activities described in this paper were funded by Ghent University, iMinds, the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), the Fund for Scientific Research-Flanders (FWO-Flanders), and the European Union.

References

[Baldwin et al.2015] 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.

[Bansal et al.2014] Mohit Bansal, Kevin Gimpel, and

Table 6: Break-down of the results on the test set. The test set contains 661 NEs. Our system detected 523 NEs of which 259 NEs were correct.

Category	Precision	Recall	F1	#Entities
company	57.89%	28.21%	37.93%	19
facility	30.77%	21.05%	25.00%	26
geo-loc	55.24%	68.10%	61.00%	143
movie	37.50%	20.00%	26.09%	8
musicartist	16.67%	2.44%	4.26%	6
other	20.78%	12.12%	15.31%	77
person	60.89%	63.74%	62.29%	179
product	22.58%	18.92%	20.59%	31
sportsteam	77.42%	34.29%	47.52%	31
tvshow	33.33%	50.00%	40.00%	3
Total	**49.52 %**	**39.18%**	**43.75%**	**523**

Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 809–815. The Association for Computer Linguistics.

[Bastien et al.2012] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

[Cano et al.2013] Amparo Elizabeth Cano, Andrea Varga, Matthew Rowe, Milan Stankovic, and Aba-Sah Dadzie. 2013. Making sense of microposts (#msm2013) concept extraction challenge. In *Making Sense of Microposts (#MSM2013) Concept Extraction Challenge*.

[Cano et al.2014] Amparo E Cano, Giuseppe Rizzo, Andrea Varga, Matthew Rowe, Stankovic Milan, and Aba-Sah Dadzie. 2014. Making sense of microposts (#Microposts2014) named entity extraction & linking challenge. In *WWW 2014, 23rd International World Wide Web Conference, Making Sense of Microposts 2014*, Seoul, South Korea.

[Collobert et al.2011] 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*, 12:2493–2537.

[Godin et al.2013] Fréderic Godin, Viktor Slavkovikj, Wesley De Neve, Benjamin Schrauwen, and Rik Van de Walle. 2013. Using topic models for twitter hashtag recommendation. In *Proceedings of the 22Nd International Conference on World Wide Web Companion*, WWW '13 Companion, pages 593–596, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

[Godin et al.2014] Fréderic Godin, Baptist Vandersmissen, Azarakhsh Jalalvand, Wesley De Neve, and Rik Van de Walle. 2014. Alleviating manual feature engineering for part-of-speech tagging of twitter microposts using distributed word representations. In *Workshop on Modern Machine Learning and Natural Language Processing (NIPS 2014)*.

[Liu et al.2011] Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. 2011. Recognizing named entities in tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 359–367, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Mikolov et al.2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

[Nadeau and Sekine2007] David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, January. Publisher: John Benjamins Publishing Company.

[Ritter et al.2011] Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1524–1534, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Srivastava et al.2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan

Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

[Tjong Kim Sang and De Meulder2003] Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

[Tkachenko and Simanovsky2012] Maksim Tkachenko and Andrey Simanovsky. 2012. Named entity recognition: Exploring features. In *11th Conference on Natural Language Processing, KONVENS 2012, Empirical Methods in Natural Language Processing, Vienna, Austria, September 19-21, 2012*, pages 118–127.

[Turian et al.2010] 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*, ACL '10, pages 384–394, Stroudsburg, PA, USA. Association for Computational Linguistics.