Impact of Data Representation on Neural Networks Performance for Named-Entity Recognition

Alodie Boissonnet
Queens' College, University of Cambridge
avmb2@cam.ac.uk

Abstract

Today, many Natural Languages Processing tasks rely on neural networks. In this report, we analyze a neural network model for the Named-Entity Recognition (NER) task, that can be viewed as a classification problem. This task was studied as part of the W-NUT Shared Task in 2017, which challenged participants with unusual and previously unseen entities. We focus on two approaches to improve the classifier performance: different text chunks representations and word embedding methods. Because of the random initialization of neural networks, we use statistical methods to compare results. We find that we achieve the best results when tagging words with an Inside/Outside scheme for named-entities and using word embeddings pre-trained on the corpus data.

1 Introduction

The task of Named-Entity Recognition (NER) consists of identifying special and unique names for specific concepts, such as persons, locations or organizations. A common method for NER is to assign labels to each word, indicating whether or not they are part of a named-entity. The W-NUT 2017 Shared Task (Derczynski et al., 2017) focuses on identifying unusual and previously-unseen entities in the context of user-generated texts, taken from Twitter, Reddit, YouTube and Stack Exchange comments. The objective is then to detect and classify novel and emerging named-entities in noisy text. Different approaches have been used to tackle the task of NER, such as Maximum Entropy Taggers (Bender et al., 2003), SVMs (Kudo and Matsumoto, 2001) or CRFs (Lafferty et al., 2001). Here, we use a neural network with long-term memory.

This task can be viewed as a tagging problem, and Tjong Kim Sang and Veenstra (1999) introduced many tagging schemes in the context of noun phrase chunks recognition. We examine the impact of these text chunks representations on performance, and we will show that two of them, IO and BILOU, outperform the others, each for different word embedding approaches.

In addition to this study on data labels, we analyze different linguistic representations. Notably, we experiment with different settings for the embedding layer of our neural network. We consider three popular word embedding methods, that are *word2vec* (Mikolov et al., 2013a), *Glove* (Pennington et al., 2014) and *FastText* (Mikolov et al., 2018). We study how word representations, obtained with these three methods or randomly initialized, affect our results. We find that the *word2vec* and *FastText* methods lead to best results when word embeddings are trained on the corpus data.

As mentioned by Reimers and Gurevych (2017), neural networks are non-deterministic methods. A single performance score is insufficient to conduct a correct comparison of different models. To avoid coming to wrong conclusions, we train 50 neural networks for each configuration set, and we study score distributions from a statistical point of view.

2 Overview of methods

2.1 Text chunks representations

The task of Named-Entity Recognition can be viewed as a tagging problem (Ramshaw and Marcus, 1995) by assigning tags to words whether or not they are part of a named-entity. Tjong Kim Sang and Veenstra (1999) proposes different tagging schemes, that differ in their treatment of the initial and final words of named-entities. We experiment with six different text chunks representations, detailed below:

- BIO1: the first word of a named-entity immediately following another one is labeled as

B, the other words of named-entities are labeled as I and words outside named-entities are labeled as O.

- BIO2: it is similar to the BIO1 scheme, but it assigns a B-tag to the first word of all named-entities.
- IOE1: the final word of a named-entity immediately preceding another one is labeled as E, the other words of named-entities are labeled as I and words outside named-entities are labeled as O.
- IOE2: it is similar to the IOE1 scheme, but it assigns an E-tag to the last word of all named-entities.
- IO: all words inside named-entities are labeled as I, and all words outside are labeled as O.
- BILOU: in a multi-token chunk, the first word is labeled as B, intermediate words are labeled as I and the last word is labeled as L. A unit-length chunk is labeled as U, and all words outside named-entities are labeled as O.

These tagging schemes vehicle different information about named-entities, and are more or less expressive. Table 1 gives an example of a sentence tags for the six tagging formats described above.

Because neural networks require fixed-length inputs, sentences are padded and an additional tag is used to capture these paddings. Table 2 gathers the number of occurrences of each tag in the different chunks representations for the training set. We notice that tag classes are highly imbalanced and we must consider it in our model.

2.2 Word embedding methods

Word representation has recently become a major topic in NLP. Before the development of *word2vec* models (Mikolov et al., 2013a), words were represented with sparse vectors of high dimension. Today, words embedding methods are able to capture morphological, semantic, contextual or even syntactic relationships in dense and low-dimensional vectors. Words that occur in similar contexts tend to have similar embedding representations. In this section, we focus on three common models: *word2vec* (Mikolov et al., 2013a), *Glove* (Pennington et al., 2014) and *FastText* (Mikolov et al., 2018). These three models provide pre-trained representations of words, as well as efficient tools to compute word embeddings with a new corpus.

word2vec We first study *word2vec* method, described in (Mikolov et al., 2013a,b,c). It uses simple neural networks to compute word embeddings based on words' context. Continuous word vectors are first learned using a simple model, and then an n-gram feedforward neural network language model is trained on the top of these distributed representations of words. In this work, we only use word vectors learned with a Continuous Bag-Of-Words (CBOW) model, whose objective is to predict a word given its context. Therefore, for a given word, it gathers the four previous and next words in a bag-of-words and gives them as input for a log-linear classifier trained to classify the current word.

Glove As seen previously, *word2vec* captures words' context, but it does not encode whether some context words appear more often than others. On the contrary, *Glove* model (Pennington et al., 2014) focuses on word statistics over the whole corpus, by combining the advantages of global matrix factorization and local context window methods. It uses a log-bilinear regression model to conduct unsupervised learning of word representations. Word vectors obtained with *GloVe* are of low dimension and reflect the probability of words' co-occurrences in the corpus.

FastText Both *word2vec* and *GloVe* methods are unable to handle unknown words. *FastText* (Bojanowski et al., 2017; Mikolov et al., 2018) aims at solving this problem by describing words as a bag of character n-grams. Therefore, a word is represented by the sum of the vector representations of its n-grams. To do so, *FastText* takes into account the internal structure of words in a Skipgram model, that predicts the context of a given word. By considering character n-grams, this embedding method is able to generalize to unknown words, as they share n-grams with other words in the corpus.

Experiments We use the *word2vec*, *Glove* and *FastText* models with four different approaches:

- We download pre-trained word representations provided by *word2vec*, *Glove* or *FastText* model, and we fix the parameters of the embedding layer of our neural network (see Section 3) with these word vectors.
- We download pre-trained word representations provided by *word2vec*, *Glove* or *FastText* model, we initialize the parameters of the

	kern	valley	will	play	the	san	diego	jewish	academy	tomorrow	at	10:00	.
BIO1	I	I	O	O	O	I	I	B	I	O	O	O	O
BIO2	B	I	O	O	O	B	I	B	I	O	O	O	O
IOE1	I	I	O	O	O	I	E	I	I	O	O	O	O
IOE2	I	E	O	O	O	I	E	I	E	O	O	O	O
IO	I	I	O	O	O	I	I	I	I	O	O	O	O
BILOU	B	L	O	O	O	B	L	B	L	O	O	O	O

Table 1: Example of sentence tags for the different tagging schemes.

Tags	B	I	E	L	U	O	padding
BIO1	26	3,115	0	0	0	59,095	292,139
BIO2	1,964	1,177	0	0	0	59,095	292,139
IOE1	0	3,115	26	0	0	59,095	292,139
IOE2	0	1,177	1,964	0	0	59,095	292,139
IO	0	3,141	0	0	0	59,095	292,139
BILOU	786	391	0	786	1,178	59,095	292,139

Table 2: Number of occurrences of each tag in the different chunks representations for the training set.

embedding layer with these word vectors, and we allow the embedding layer to update them during training.

- We train a new *word2vec* or *FastText* model with our training dataset, and we fix the parameters of the embedding layer of our neural network with these word vectors.
- We train a new *word2vec* or *FastText* model with our training dataset, we initialize the parameters of the embedding layer with these word vectors, and we allow the embedding layer to update them during training.

We also study a model without any pre-trained word embedding vectors. Table 3 summarizes all experiments conducted.

3 Model description

The model used to solve this NER task is an LSTM neural network (Hochreiter and Schmidhuber, 1997; Lample et al., 2016) made of four layers: an embedding one whose size depends of the word embedding method chosen (see Table 3), followed by a bi-directional LSTM of 50 units, then a dropout layer with a rate of 0.5 and a final dense layer with softmax activation. An early stopping criterion is set to halt training if no improvement has been seen after 10 epochs. We compile our model with an Adam optimizer and use a cross-entropy categorical loss for training. The implementation of this neural network is available in the notebook provided with this report.

Because neural networks rely on a random initialization (Reimers and Gurevych, 2017), we conduct a statistical comparison of our models. To do so, we train each model 50 times, and we analyze their global performance to determine the best one. Therefore, speeding up the training of models appears to be necessary to do this work in a limited time. For this reason, we use a GPU provided by Google Colaboratory, that divides the training time of each model by about 40. However, using GPU forces the LSTM layer to have a recurrent dropout rate set to 0.

4 Data

4.1 Sources and splits

This task on NER uses the data provided by the W-NUT 2017 Shared Task (Derczynski et al., 2017). The training set is made of 1,000 annotated tweets, totaling 65,124 tokens. The development set comprises 1,008 YouTube comments from the all-time top videos (15,734 tokens), and the test set contains 1,287 comments from Twitter, Reddit and StackExchange (23,394 tokens). All these sources of data gather many unusual and previously-unseen entities, as desired by the Shared Task.

The datasets provided by the Shared Task are already preprocessed to make their use easier. Notably, data are filtered for common entities to make sure that the development and test sets don't share any entities with the training set. Texts are also tokenized, so that each line of the dataset contains only one word.

Embedding method	Pre-trained vectors	Training data	Embedding size	Embedding updates
No embedding	/	/	128	Yes
word2vec	Yes	Google News	300	No
word2vec	Yes	Google News	300	Yes
word2vec	No	training set	300	No
Glove	Yes	Twitter	25	No
Glove	Yes	Twitter	200	No
Glove	Yes	Twitter	200	Yes
FastText	Yes	Wiki News	300	No
FastText	Yes	Wiki News	300	Yes
FastText	No	training set	300	No

Table 3: Summary of the experiments conducted in this work.

4.2 Preprocessing

As for any Natural Language Processing application, it is necessary to preprocessed data to adapt them to our task and model. In this work, we first start by converting NER labels of the training set into the tagging scheme we are experimenting with (see Section 2.1), before encoding word tokens and NER labels as integer values. For word tokens, we create a case-sensitive index of all words and use it to convert word tokens. We assign a special integer value to out-of-vocabulary words. In order to use the word embedding methods describe in Section 2.2, we also convert word tokens to lower case. Then we transform our table-format data into text sequences, where each sequence contains one tweet, post or comment, and we pad all these sequences to make them of the length of the longest sequence. A new NER tag is created to deal with paddings. The final step consists of encoding NER label sequences with a one-hot scheme and weighting these labels to overcome the imbalanced classes issue: weights are inversely proportional to class occurrences in order that common classes have a lower weight.

Predictions obtained with the model are also processed before evaluation. Notably, we remove the padding from sequences of predictions and transform them into a table format. The last step is to convert predictions into the BIO2 scheme, as it the one used to label the development and test sets. When experimenting with the BIO2 scheme, we also correct predictions to get rid of any inconsistency. Thus, a named-entity starting with an I-label is converted into a B-label. Likewise, the number of occurrences of the tag B in the BIO1 scheme (see Table 2) shows that it is very rare to have a named-entity immediately following another one: we choose to convert all B-tags that are not at the beginning of a named-entity into I-tags.

5 Evaluation method

In this work, we use a unique evaluation measure, an entity F1 score, which is the harmonic mean of the entity-level precision and recall. Contrary to a surface F1 score, this evaluation method considers whole named-entities labels rather than token ones. Therefore, a correctly labeled multi-token entity, such as *Empire State Building* is rewarded only once with an entity F1 score, whereas it would be rewarded three times with a surface F1 score.

Because of the random initialization of neural networks (Reimers and Gurevych, 2017), we examine the global performance over 50 experiments of all models considered in this work. To do so, we compare the mean and standard deviation of the F1 scores obtained for each model.

6 Results and discussion

We report in Table 4 and Figure 2 the performances achieved for each pair of tagging scheme and word embedding method. These results consist of the mean and the standard deviation of the 50 F1-scores obtained for all experiments. Other figures illustrating results and comparing methods can be found in the appendix (see Appendix A).

Tagging schemes We first show that the IO representation outperforms the other chunk representations, except if words are encoded with the *Glove* method. In that case, the BILOU representation achieves the best performance. This conclusion is surprising as the IO and BILOU schemes are the two most different chunk representations. While IO is the least expressive representation and specifies only if a word is part of a named-entity, BILOU

Model	Pre-trained	Updates	BIO1		BIO2		IOE1		IOE2		IO		BILOU	
			mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
No embedding	/	Y	0.24	0.03	0.28	0.04	0.24	0.03	0.24	0.03	**0.30**	0.03	0.20	0.04
word2vec	Y	N	0.09	0.02	**0.13**	0.02	0.11	0.02	**0.13**	0.03	**0.13**	0.02	**0.13**	0.02
word2vec	Y	Y	0.14	0.02	0.21	0.04	0.17	0.03	0.21	0.04	**0.24**	0.05	0.17	0.04
word2vec	N	N	0.20	0.02	0.20	0.01	0.20	0.02	0.18	0.01	**0.23**	0.02	0.16	0.01
word2vec	N	Y	0.31	0.03	**0.34**	0.01	0.32	0.02	0.30	0.02	**0.34**	0.02	0.32	0.01
GloVe 25	Y	N	0.15	0.02	0.17	0.03	0.16	0.01	0.16	0.02	0.17	0.02	**0.20**	0.03
GloVe 25	Y	Y	0.21	0.03	0.26	0.05	0.24	0.04	0.26	0.04	0.27	0.05	**0.30**	0.03
GloVe 200	Y	N	0.19	0.02	0.22	0.02	0.20	0.02	0.21	0.02	0.21	0.02	**0.24**	0.02
GloVe 200	Y	Y	0.21	0.02	0.25	0.02	0.22	0.02	0.25	0.03	0.25	0.02	**0.26**	0.03
FastText	Y	N	0.23	0.04	0.27	0.03	0.25	0.05	0.25	0.02	**0.28**	0.03	0.27	0.03
FastText	Y	Y	0.19	0.03	0.23	0.04	0.20	0.03	0.23	0.02	**0.28**	0.03	0.20	0.05
FastText	N	N	0.18	0.01	**0.19**	0.01	**0.19**	0.01	0.15	0.01	**0.19**	0.01	0.14	0.01
FastText	N	Y	0.30	0.02	0.34	0.01	0.33	0.02	0.30	0.01	**0.35**	0.01	0.31	0.01

Table 4: Mean and standard deviation of the F1-scores obtained for each model.

is the most expressive representation, encodes if a word is at the beginning, in the middle, or at the end of a named-entity, and specifies whether or not a named-entity is multi-token. Therefore, models with *GloVe* embeddings achieve better results when they have more information about named-entity. We can also notice that these models coupled with a BIO2 or IOE2 tagging scheme outperform the ones coupled respectively with a BIO1 or IOE1 scheme, which is consistent with the previous conclusion about *Glove* embeddings.

However, it is predictable that *word2vec* and *FastText* embedding methods perform best with the same tagging scheme, as they rely on a very similar method. On the contrary, *GloVe* method uses a completely different approach to encode words, using statistics over the whole corpus.

The previous conclusion about the good performances achieved with the IO scheme and the results presented in Table 4 for the models with *word2vec* and *FastText* methods contrast with Ratinov and Roth (2009) work on NER models. In their paper, they state that the BILOU scheme significantly improves the performance of a feature-based NER tagger. However, this conclusion is not conflicting with this work, but it shows that the choice of the tagging scheme depends of the choice of the model, whether a neural network or a feature-based one.

In addition, the BIO2 scheme is by far the most commonly used text chunks representation. This work, combined with (Ratinov and Roth, 2009), suggests that this tagging scheme does not achieve the best results, even though Figure 1 shows that it performs well on the overall. This figure represents the average F1-score obtained for each tagging scheme across all models.

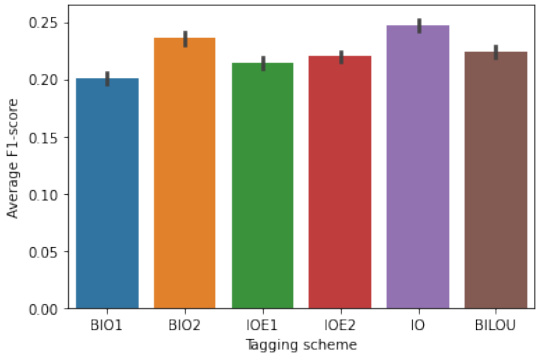

Figure 1: Average F1-scores obtained for each tagging scheme with a 95% confidence interval.

Word embedding methods To begin with, the comparison of the performance of the models with a *GloVe* embedding method reveals that a larger embedding size improves results. In this work, we use two different embedding sizes, 25 and 200, to construct *GloVe* word vectors. Whether or not the embedding layer is allowed to be updated during the training of the model, larger word vectors lead to better results. However, this embedding size can not be increased indefinitely. The purpose of an embedding method is precisely to limit the dimension of word vectors. For this reason, we only consider word vectors with a dimension of 300 for the other embedding methods.

Concerning the choice of word embedding methods, no particular embedding method significantly outperforms the others, with the exception of two of them detailed below. We even notice that a model without pre-trained word representations performs as well as, or even better than, the others. This shows that word embedding methods are not very efficient for a NER task on unusual and previously-unseen named-entities. This conclusion was pre-

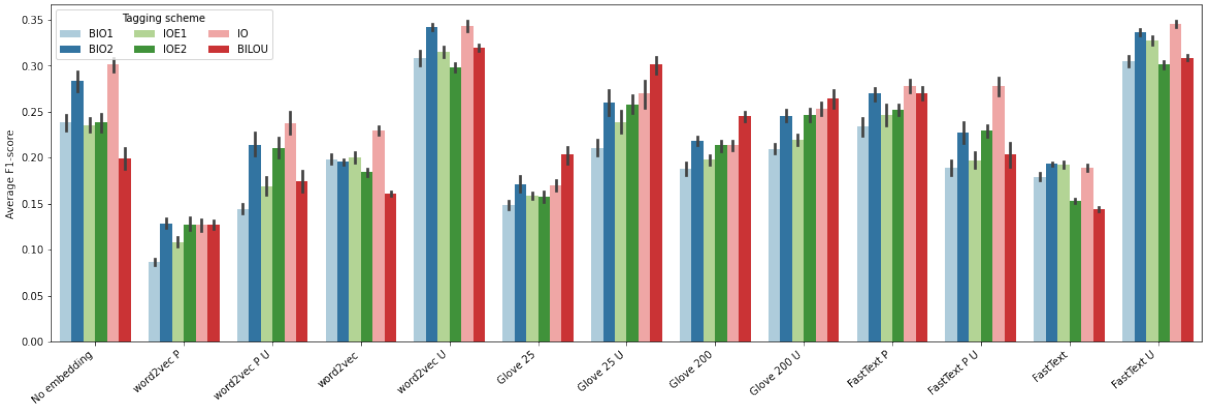

Figure 2: Mean of the F1-scores obtained for each model with a 95% confidence interval. P means that the model uses a pre-trained word embedding method, and U means that the embedding layer is updated during training.

dictable: word embedding methods struggle to encode rare words. More precisely, when words are encoded with the pre-trained models described in Section 2.2, about a third of words are represented by a null vector. Indeed, the data used for this task had been chosen because they contain unusual words. Even if we allow the parameters of the embedding layer to be updated during the training of the neural network, we lose a lot of information about all rare words in our data, and the neural model is not able to recover all words' meaning during training. Even the *FastText* method, which is supposed to be better at representing rare words, fails at representing words in this dataset. This may be explained by the distinct characteristic of user-generated text that contains many abbreviations, spelling errors and unusual characters.

However, Figure 2 shows that two models outperform the others. In these two models, the parameters of the embedding layer are initialized with word embeddings obtained by training *word2vec* or *FastText* methods with the training dataset, and are allowed to be updated during the training of the NER neural network. These results are not surprising: pre-trained word embedding methods are not able to encode rare words, but we can encode them by training these embedding methods on our dataset. Yet, the very small size of our training dataset is detrimental to the quality of word embeddings. For this reason, when preventing these word embeddings to be updated during training, models are not able to achieve good results. On the contrary, when word embeddings are allowed to be updated, the neural network better captures the meaning and context of words and performs well.

This last conclusion suggests that it could be interesting to use pre-trained word embeddings from methods such as *word2vec*, *GloVe* or *FastText*, and to train again these word vectors with the dataset we work on. It may help to capture the meaning and context of common words while encoding the unusual words of the dataset.

Final model Our final model uses the IO scheme to tag named-entities, pre-trains word embeddings with the *FastText* method and allows the embedding layer to be updated during training. We achieve an F1-score of 0.264 on the test set, which suggests that improvements can still be done. Indeed, the best teams of the W-NUT 2017 Shared Task obtained an F1-score higher than 0.40. Other neural network configurations or a feature-based model are tracks to consider.

7 Conclusion

In this work, we presented a neural network model to solve a Named-Entity Recognition task on unusual and previously-unseen named-entities in user-generated texts. We explored different text chunks representations and word embedding methods. We showed that models with no pre-trained embedding method or using the *word2vec* or *FastText* embedding method achieve higher performance when they use the IO scheme to tag named-entities. On the contrary, models with a *Glove* embedding method perform better with the more expressive BILOU scheme. In addition, best results have been obtained when word embeddings are directly trained on our dataset with *word2vec* or *FastText* method. This approach allows models to better encode rare words, that are over-represented in our data.

References

Oliver Bender, Franz Josef Och, and Hermann Ney. 2003. Maximum entropy models for named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 148–151.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147, Copenhagen, Denmark. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Second Meeting of the North American Chapter of the Association for Computational Linguistics*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

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

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Neural and Information Processing System (NIPS)*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

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 (CoNLL-2009)*, pages 147–155, Boulder, Colorado. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348, Copenhagen, Denmark. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Jorn Veenstra. 1999. Representing text chunks. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 173–179, Bergen, Norway. Association for Computational Linguistics.

A Additional graphs

This appendix gathers two figures that illustrate the results obtained in this work. The first one represents the average F1-score obtained for each tagging scheme, and is split by word embedding method. We can see that models with a *Glove* embedding approach performs better with the BILOU tagging scheme. Likewise, all other models perform better with the IO one. The BIO2 scheme achieves also good results on the overall.

The second figure shows the average F1-scores obtained for each word embedding method, across all models. We can clearly notice that models that train word embeddings on the corpus data with *word2vec* or *FastText* methods outperform the others. Models with no embedding method achieve also very good performance, which supports the previous conclusion stating that pre-trained word embedding methods struggle to encode the unusual words of our dataset.

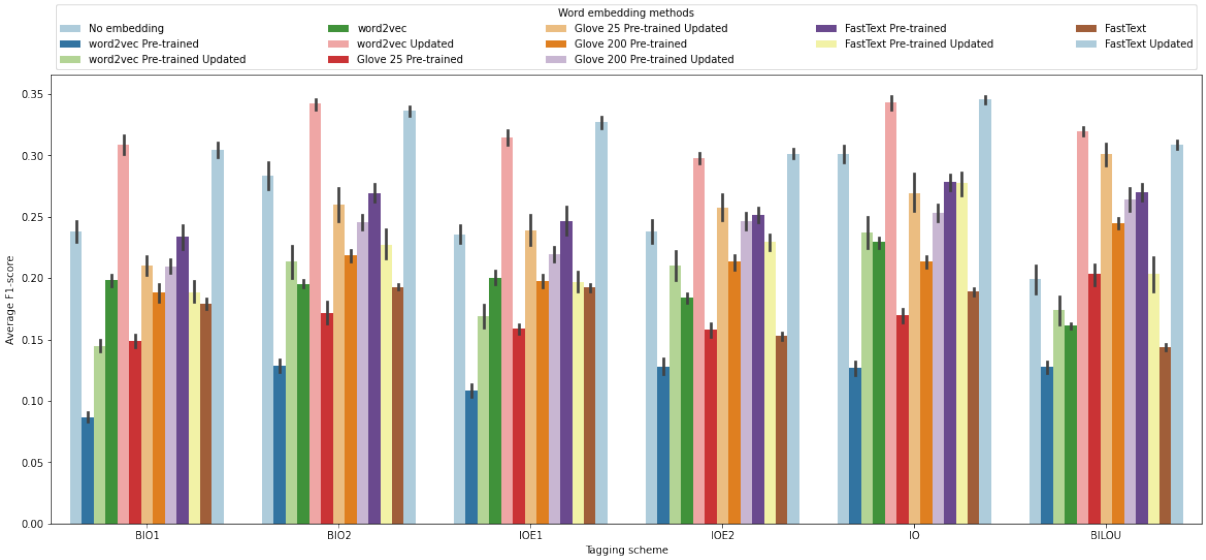

Figure 3: Mean of the F1-scores obtained for each tagging scheme with a 95% confidence interval. P means that the model uses a pre-trained word embedding method, and U means that the embedding layer is updated during training.

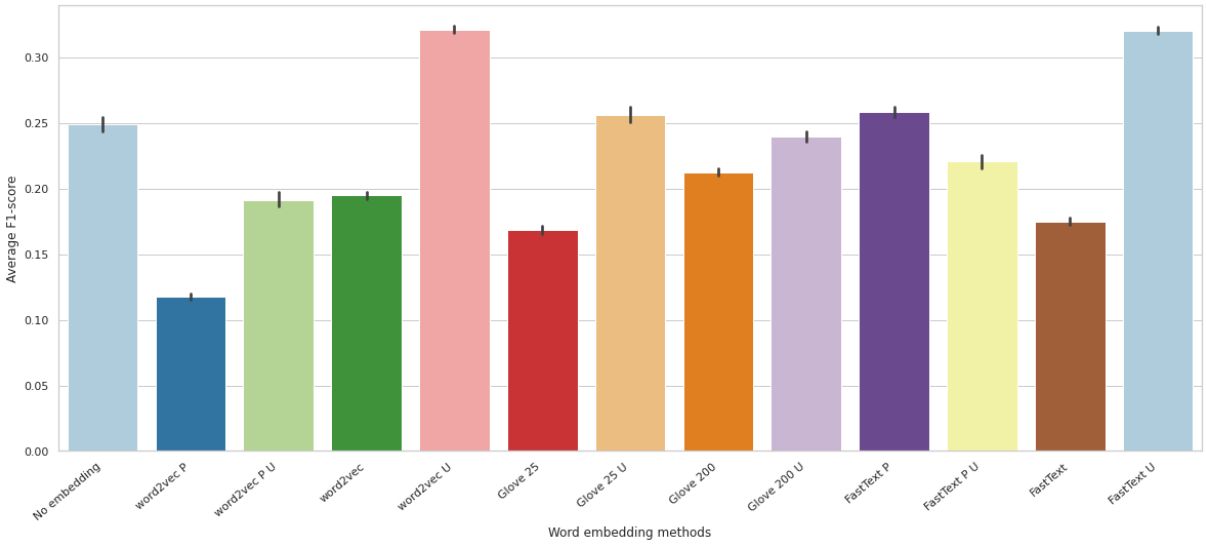

Figure 4: Average F1-scores obtained for each word embedding method across all models with a 95% confidence interval.