First assessment of Graph Machine Learning approaches to Portuguese Named Entity Recognition

Gabriel Silva
IEETA / LASI, DETI - UA
grsilva@ua.pt

Mário Rodriges
IEETA / LASI, ESTGA - UA
mjfr@ua.pt

António Teixeira
IEETA / LASI, DETI - UA
ajst@ua.pt

Marlene Amorim
GOVCOPP, DEGEIT - UA
mamorim@ua.pt

Abstract

Currently there are several methods to annotate different levels of a document, however, these methods all have their own output and some even create their own formats to share the results of processing. This makes it so that retrieving, sharing, and comparing information from these different methods is not a trivial task. Knowledge graphs are a flexible tool that can be used to counter this difficulty as it creates the possibility of having annotations at different levels of the text (document, sentence and word for example). Besides this, Knowledge Graphs also provide us with the possibility of using different Machine Learning algorithms which can be applied to different Natural Language Processing tasks, such as Named Entity Recognition.

In this work we present a first assessment of using Graph Machine Learning algorithms to perform Named Entity Recognition on the Portuguese Language. We use the Portuguese portion of the WikiNER dataset and process it as a Knowledge Graph with extra features from Universal Dependencies to perform a Node Classification Task for Named Entity Recognition. We present the results for 3 different GraphML approaches with different sub-graph combinations and discuss how this could be used in the future to predict new nodes that come into the network. The approach used can be adapted to other languages as there is nothing specific to the Portuguese language other than the dataset.

1 Introduction

With the the expansion of the internet and IoT, the world saw a dramatic increase in the amount of data that is generated every day (Hilbert, 2016) from various sources in different formats. However, this data can become useful information, for example, twitter, can be used to identify adverse drug reactions (Cocos et al., 2017) or analyse comments to have a better understanding of patient feedback (Khanbhai et al., 2021).

With all these different sources of data, having a format that can provide support to different processing methods is crucial. Knowledge graphs are a flexible format that can accommodate all the differences in these sources. These graphs can accommodate different annotations at different levels of the documents and are able to be integrated in a vast, already existing, semantic web ecosystem. To turn this data into information we still need to apply Natural Language Processing (NLP) techniques such as Named Entity Recognition (NER) and Relation Discovery (RD). In the past few years the field of NLP has seen a big leap forward thanks to the appearance of models such as Convolutional Neural Networks (CNNs) (Krizhevsky et al., 2012) and Bidirectional Long Short-Term memory (Bi-LSTMs) (Lample et al., 2016) and, more recently, the use of pre-trained models, such as BERT (Devlin et al., 2019) or BART (Lewis et al., 2020), coupled with the others techniques further improved the state of the art. However, as the authors of (Battaglia et al., 2018) noted, in order for these models to improve even further it is necessary to be able to generalize beyond their experiences, the current models rely on relational assumptions to make correct predictions. This is where the use of Graphs and GraphML can be used to improve the field (Battaglia et al., 2018). These methods can handle a wide range of problems and data types and can even be merged with the previous techniques. Several works have already explored Graph Networks by themselves for NLP tasks or by merging them with other Deep Learning (DL) techniques (Carbonell et al., 2021; Cetoli et al., 2017; Madan et al., 2023) in different fields.

In this work we perform a first assessment of Graph ML techniques for Portuguese Named Entity Recognition (NER). We process the Portuguese part of the WikiNER (Nothman et al., 2013) dataset with Universal Dependencies (UD) (de Marneffe

et al., 2014) annotations by using OntoUD (Silva et al., 2023) and apply three different Graph ML algorithms to the dataset, Graph Convolutional Networks (GCNs) (Kipf and Welling, 2017), Graph Attention Networks (GATs) (Veličković et al., 2018) and GCNs coupled with DeepGraphInfomax (Veličković et al., 2018). We attempt several data splits and sub-graphs to see how the algorithms perform in data-scarce environments and present our findings.

The rest of the paper is structured as follows: Section 2 will describe the methodology used to build the testing of these different algorithms and the preparation of the dataset. Section 3 we will report the results achieved in each test that was performed and highlight the best results. Section 4 will focus on the conclusions achieved in this work as well as what can be done in the future to further assess the suitability of Graph ML for Portuguese NLP.

2 Methodology

This section will focus on the methodology taken for preforming the different test on each algorithm. We will talk about how the dataset was built, the different features of the edge nodes that will be used to predict entities and the general testing methodology.

2.1 Dataset Description and Processing

As was said in Section 1 the dataset used was the Portuguese portion of WikiNER (Nothman et al., 2013) with Universal Dependency (de Marneffe et al., 2014) tags in it. The WikiNER dataset was processed by OntoUD (Silva et al., 2023), this tool takes care of the UD annotations and the NER entity annotations and converts it into a Knowledge Graph which was then uploaded onto Virtuoso[1]. However, the dataset cannot be used directly from Virtuoso. Using SPARQL [2] queries we fetch the WikiNER sentences and all of their dependents (the words) and build a rdflib [3] graph which we can convert into features and targets with the help of StellarGraph (Data61, 2018). This is also the library that was used for the algorithms.

Most of the features used are the edges on each of the "Word" nodes from each sentence, namely: type, word, poscoarse, pos, lemma, id, feats, edge, depGraph, previousWord, head, senttext, fromText, nextSentence, fromSentence. With the targets being the edge "wikinerEntity" which helps us identify 4 different entity types as well as a target for when a node is not an entity: "No", "PER", "LOC", "MISC" and "ORG". These features are described more in-depth in the OntoUD paper (Silva et al., 2023).

Sentence ID sub-graph	No	PER	LOC	MISC	ORG	Total
1-300	405	391	295	59	50	1200
301-600	364	174	647	88	45	1318
1501-1800	373	145	452	130	82	1182
2123-2422	361	408	603	57	39	1468
Random 300	358	316	560	62	77	1373

Table 1: Entity count for each WikiNER sentence list that was tested.

Since WikiNER is a big dataset it is not feasible to use the entirety of the dataset to train these algorithms, as such, we create different sub-graphs based on 300 different sentences. We try different combinations of sentences to see how robust these algorithms are both during training and testing. These sub-graphs are then split into training, test and dev, using a static seed to keep the same split across the algorithms, with the results reported being the ones from the test set. The split followed a stratified approach since the dataset is imbalanced. We also reduced the number of "No" that was present in the dataset as not to have an overwhelmingly percentage of the dataset have no entities and attempt to have a better balance between words that are not entities and words that are. Table 1 will show the number of elements for each class in each of the sub-graph group that was used.

As we can see despite always fetching 299 sentences the number of nodes to classify is different. This is due to sentences having a variable number of words so we cannot expect the same number of nodes. There was no criteria to picking these ranges of sentences other than to have an ample sample from different points in the dataset and see how the algorithm performs for these different ranges. For each of these sub-graphs we tried 5 different train/test/dev splits to see how the model performs with less data. The values for train test and dev were the following: First split - 80% - 4% - 16%, Second split - 70% - 9% - 21%, Third split - 50% - 25% - 25%, Forth split: 30% - 49% - 20% and Fifth split - 20% - 64% - 16%.

The goal is to test these algorithms in data scarce environments as well as environments where there

[1] https://virtuoso.openlinksw.com/
[2] https://www.w3.org/TR/sparql11-query/
[3] https://rdflib.readthedocs.io/en/stable/

is a lot of data available for training and check their performance.

2.2 Testing

The testing was done with the StellarGraph (Data61, 2018) library. This is a library that is built on top of Keras (Chollet et al., 2015) to simplify the pipeline of creating Graph ML algorithms. We chose three different algorithms to test: Graph Convolutional Networks, Graph Attention Networks and DeepGraphInfomax with a final GCN prediction layer. We kept the parameters the same for every test to maintain consistency between tests. For the GCN the parameters were the following: 2 layers with size 16 with ReLU activation, 0.4 dropout, ADAM optimizer with a learning rate of 0.01 and categorical cross-entropy as our loss function. For the GAT the parameters were as follows: 2 layers of size 8 and number of targets with elu and softmax activation respectively, in and attention dropout at 0.5, Adam optimizer with a learning rate of 0.005 and categorical cross-entropy loss function.

All the testing was done with 500 epochs with an early stop condition and the dataset splits are mentioned in Section 2.1. The early stop condition is tied to the accuracy on the validation set with the patience parameter set to 50 for all three models. The idea behind splitting the data into 5 sub-graphs each with subsequently less training data was to monitor how these models perform in environments that do not have a lot of training data available to them.

Since the DeepGraphInfomax (Veličković et al., 2018) is an unsupervised algorithm whose goal is to learn node representations within a graph we paired it with a GCN to form a semi-supervised algorithm and see if this pairing would improve our results significantly. The DeepGraphInfomax model is described in the original paper (Veličković et al., 2018) and the GCN model is the same as the standalone model. We train the DeepGraphInfomax on our graph for 500 epochs and then use it as a pretrained model for the GCN.

3 Results

This section will discuss the results obtained with the dataset and algorithms that were mention in Section 2.

3.1 GCN

This was the first model to be tested due to it being the more simple model and the basis for the last model that was tested. The parameters used for this testing are described in Section 2.2. Despite being able to train for 500 epochs these models only trained between 120 to 250 epochs depending on the split and the amount of data the model had available. Figure 1 shows an example of the training losses and accuracy but for different data splits. These curves are pretty similar and the same pattern can be found in the rest of the training curves for the remainder of the splits with the different data.

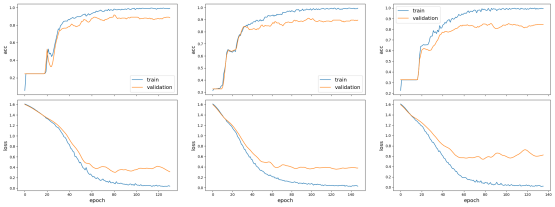

Figure 1: Example of a training curve for the GCN algorithm with the three first datasplits. The top graphs show the training accuracy and the bottom graphs the loss. From left to right we have the First split, the Second split and the Third split.

The best result achieved by the model was an accuracy of 92.5% with a loss of 0.35. This was achieved in the 301-600 sub-graph with the First split (80/4/16), however for this same sub-graph training with 10% less data, the second split (70/9/21) achieved a comparable result of 92.4% accuracy.

3.2 GAT

The Graph Attention Network was the one that performed the worse by a decent margin. This model is probably one that would require a more complex network and fine-tuning in order to achieve better results.

The difference from the best performing GAT model to the best split GCN model is an accuracy of 3.77% both for an 80/20 data split. In Figure 2 we can see that the training is a lot more hectic when it comes to accuracy and loss than the one done by the GCNs.

Some of these models also ended their training a lot sooner than the GCN with an epoch range between 60 and 270. The model that performed best here was with a First split (80/4/16) with an accuracy of about 88.14%, however, for the same

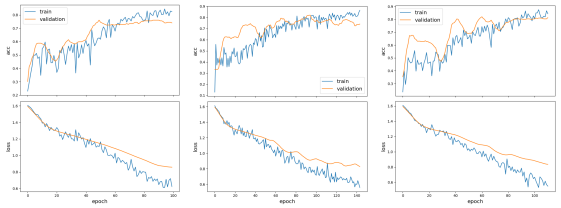

Figure 2: Example of a training curve for the GAT algorithm with the 0-300 sub-graph with the three first datasplits. The top graphs show the training accuracy and the bottom graphs the loss. From left to right we have the First split, the Second split and the Third split.

sub-graph a Third split (50/25/25) model managed comparable results with 87.77% accuracy.

3.3 DeepGraphInfomax + GCN

This was the more complex model but it still fell short to the GCN. In order to perform the testing of this model there were two training steps. The unsupervised training of the DeepGraph Infomax network and then the training of the GCN using the previously trained model. The best result for this model was using the second data split (70/9/21) with the 2123-2422 sub-graph with an accuracy of 95.49%. Figure 3 shows the training loss and accuracy for three different splits in the same sub-graph.

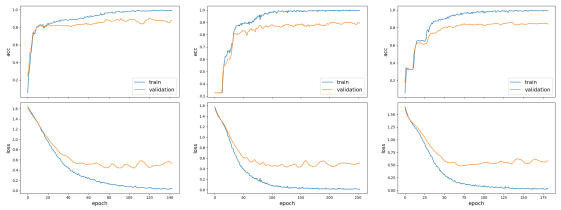

Figure 3: Example of a training curve for the DeepGraphInfomax/GCN algorithm with the three first datasplits. The top graphs show the training accuracy and the bottom graphs the loss. From left to right we have the First split, the Second split and the Third split.

The discrepancy in accuracy when training with 80% and 20% of the data is consistent with the results shown by the other models. Table 2 shows the mean results for each data split. The model had results close to the GCN, however, the extra training step made it take a lot longer to train.

Table 2 shows the mean accuracy achieved by each model in each of the data splits.

4 Conclusion and Future Work

This paper presents a first assessment of using Graph ML algorithms to perform NER for the Portuguese Language. The results are good, with the best model achieving a mean accuracy of 89.62% over all the sub-graphs and performing well on data scarce environments with a difference of about 11% in accuracy between the best model, which was the GCN. The others models preformed comparatively, but the GAT algorithm was lagging behind with their best result tying the worst result of GCNs.

	First Split	Second Split	Third Split	Fourth Split	Fifth Split
GCN	**0.8962**	0.8673	0.85469	0.8339	0.7892
GAT	0.7839	0.7928	0.7892	0.7334	0.6987
DGI+GCN	0.8858	0.8877	0.8231	0.7730	0.7780

Table 2: Mean accuracy results for the different sub-graphs using different algorithms.

One of the limitations of some of these methods is the need to have the whole Graph available to them at training time, even if they will not use every node to train, in order to generalize to other nodes. This means that whenever a new node comes in we have to re-train the whole graph just to be able to classify that node. This leads to huge resources being needed when the graphs start getting big and lots of nodes get added.

From the results achieved we can see that Graph ML is promising for Portuguese NLP tasks. The approach was used for Named Entity Recognition, however, it can be used for different tasks making it promising and worth exploring even further. For future work we intend to apply and extend this work to different, more recent, Graph ML algorithms such as, for example, Graphormers (Ying et al., 2021).

We also intend to try different Graph ML alternatives, ones that allow for inductive representation, for example, GraphSAGE (Hamilton et al., 2017) to overcome this limitation. Another option is to find the best data split with the least amount of data possible so that whenever new nodes come in we can train the model with this constant set of data and then perform prediction only on the new nodes that come in. Another option is to look into fine-tuning the parameters of these networks to achieve better results.

Acknowledgements

This research is funded by National Funds through the FCT - Foundation for Science and Technology (UI/BD/153571/2022). It is also funded, Research Unit IEETA, by National Funds through the FCT - Foundation for Science and Technology, in the context of the project UIDB/00127/2020.

References

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Manuel Carbonell, Pau Riba, Mauricio Villegas, Alicia Fornés, and Josep Lladós. 2021. Named entity recognition and relation extraction with graph neural networks in semi structured documents. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 9622–9627.

Alberto Cetoli, Stefano Bragaglia, Andrew O'Harney, and Marc Sloan. 2017. Graph convolutional networks for named entity recognition. In *Proceedings of the 16th International Workshop on Treebanks and Linguistic Theories*, pages 37–45, Prague, Czech Republic.

François Chollet et al. 2015. Keras. `https://keras.io`.

Anne Cocos, Alexander G Fiks, and Aaron J Masino. 2017. Deep learning for pharmacovigilance: recurrent neural network architectures for labeling adverse drug reactions in Twitter posts. *Journal of the American Medical Informatics Association*, 24(4):813–821.

CSIRO's Data61. 2018. Stellargraph machine learning library. `https://github.com/stellargraph/stellargraph`.

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4585–4592, Reykjavik, Iceland. European Language Resources Association (ELRA).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Martin Hilbert. 2016. Big Data for Development: A Review of Promises and Challenges. *Development Policy Review*, 34:135–174.

Mustafa Khanbhai, Patrick Anyadi, Joshua Symons, Kelsey Flott, Ara Darzi, and Erik Mayer. 2021. Applying natural language processing and machine learning techniques to patient experience feedback: a systematic review. *BMJ Health Care Inform.*, 28(1):e100262.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

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

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Monica Madan, Ashima Rani, and Neha Bhateja. 2023. Applications of named entity recognition using graph convolution network. *SN Computer Science*, 4(3):266.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R Curran. 2013. Learning multilingual named entity recognition from wikipedia. *Artificial Intelligence*, 194:151–175.

Gabriel Silva, Mário Rodrigues, António Teixeira, and Marlene Amorim. 2023. A Framework for Fostering Easier Access to Enriched Textual Information. In *12th Symposium on Languages, Applications and Technologies (SLATE 2023)*, volume 113 of *Open Access Series in Informatics (OASIcs)*, pages 2:1–2:14, Dagstuhl, Germany. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep graph infomax. *arXiv preprint arXiv:1809.10341*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888.