Using Embeddings to Improve Named Entity Recognition Classification with Graphs

Gabriel Silva
IEETA, DETI, University of Aveiro, Portugal
LASI – Intelligent System Associate Laboratory, Portugal

Mário Rodrigues
IEETA, ESTGA, University of Aveiro, Portugal
LASI – Intelligent System Associate Laboratory, Portugal

António Teixeira
IEETA, DETI, University of Aveiro, Portugal
LASI – Intelligent System Associate Laboratory, Portugal

Marlene Amorim
GOVCOPP, DEGEIT, University of Aveiro, Portugal

Abstract

Richer information has potential to improve performance of NLP (Natural Language Processing) tasks such as Named Entity Recognition. A linear sequence of words can be enriched with the sentence structure, as well as their syntactic structure. However, traditional NLP methods do not contemplate this kind of information. With the use of Knowledge Graphs all this information can be represented and made use off by Graph ML (Machine Learning) techniques. Previous experiments using only graphs with their syntactic structure as input to current state-of-the-art Graph ML models failed to prove the potential of the technology. As such, in this paper the use of word embeddings is explored as an additional enrichment of the graph and, in consequence, of the input to the classification models. This use of embeddings adds a layer of context that was previously missing when using only syntactic information.

The proposed method was assessed using CoNLL dataset and results showed noticeable improvements in performance when adding embeddings. The best accuracy results with embedings attained 94.73 % accuracy, compared to the 88.58 % without embedings while metrics such as Macro-F1, Precision and Recall achieved an improvement in performance of over 20%. We test these models with a different number of classes to assess whether the quality of them would degrade or not. Due to the use of inductive learning methods (such as Graph SAGE) these results provide us with models that can be used in real-world scenarios as there is no need to re-train the whole graph to predict on new data points as is the case with traditional Graph ML methods (for example, Graph Convolutional Networks).

2012 ACM Subject Classification Information systems → Document representation; Information systems → Ontologies

Keywords and phrases Knowledge graphs, Enriched data, Natural language processing, Named Entity Recognition

Digital Object Identifier 10.4230/OASIcs.SLATE.2024.1

Supplementary Material
Software (Source Code): `https://github.com/gabrielrsilva11/GraphBuilderAPI`

Funding *Gabriel Silva*: This research is funded by National Funds through the FCT – Foundation for Science and Technology (UI/BD/153571/2022) and by the RCNA project FCT/CPCA-IAC/2023/01. The, Research Unit IEETA, is also funded by National Funds through the FCT – Foundation for Science and Technology, in the context of the project UIDB/00127/2020.

1 Introduction

With richer information having the potential to improve performance of NLP (Natural Language Processing) tasks it is important to have data in a format that can support different processing methods and several types of annotations. Knowledge graphs are one of these formats. They are a flexible format that can accommodate processing at different levels of the document (document, text and sentence level) and their respective annotations as well as keep connections between words and sentences intact.

Despite this fact, in the past few years all the developments in the NLP field have been to traditional methods of processing. We've had the appearence of models such as Convolutional Neural Networks (CNNs) [10] and Bidirectional Long Short-Term memory (Bi-LSTMs) [12] and, more recently, the use of pretrained models, such as BERT [6] or BART [14], coupled with others techniques, for example Word Embeddings [16], further improved the state of the art. For more general use Large Language Models and Generative AI such as ChatGPT [18] have taken both the academic world and the general public by storm. However, despite all these progresses the authors of [1] noted that in order for these models to further improve they need to be able to generalize beyond their experiences. As these models rely on relational assumptions to make correct predictions this is where Graph Machine Learning (Graph ML) can be used to take the field a step further. These methods can handle a wide range of problems and data types and even utilize the work developed by the more "traditional" techniques [4]. Graph ML techniques have already been explored for NLP and, more specifically, for NER (Named Entity Recognition, extracting entities from a text) applications for some time achieving good results [2, 3, 15, 21].

Another common problem that graph models suffer when it comes to real world applications is the fact that most of them are performed using a Transductive Learning approach. What this means is that whenever a new data point comes the whole model has to be retrained which is not feasible with the rate at which data is generated[23].

As such these are the problems we have currently identified: (1) Current models, in general, when processing textual data lose information about connections between words (2) Using only syntactic information in a graph is not enough to achieve relevant results and (3) Graph ML methods utilize, in general, transductive methods where you need to re-train the whole graph to contemplate new data. Keeping these problems in mind we set two objectives for this work: (1) Explore adding Embedding information to the syntactic information in a Graph to improve results and (2) Create models that use an inductive learning approach, for example, GraphSAGE [8] so that they are more viable to implement in real-world applications.

As use case for our experiments a NER task was selected. To pursue our objectives we adopted the CoNLL-2003 [22] dataset and tested with a different number of classes to assess whether the quality of results degrades or not.

The paper is structured in three different sections, starting off by describing the Methods (Section 2) used: preparation of the dataset, creating the graph, applying embeddings to the graph and describing the models and the training. Following this section we present the results, on Section 3, and present our Conclusions and Future Work, on last Section, the 4th.

2 Method

Taking in consideration the objectives defined above, the CoNLL-2003 [22] dataset was processed in two different ways. The first one we use the dataset as is with all of the tags for entities. The second way we trim tags, going from 8 classes to 5, by excluding boundaries information from the dataset and process it the same way as we did the first one.

To assess the possible effects of the size of the set of entities considered, experiments were performed for two different numbers of classes.

This section presents information regarding: (1) dataset preparation; (2) process of graph creation from text; (3) which nodes and edges are used as features; (4) how the embeddings are generated; and (5) the models used for NER and how they were trained.

2.1 Dataset Preparation

As previously mentioned in the introduction, the dataset used in this work is the CoNLL [22] dataset. This dataset was chosen due to its accessibility, making it easy to replicate the work here presented, as well as the additional Chunking and POS (Part-of-Speech) tagging information that can be used as attributes in our graph.

This is a NER dataset which was a part of CoNLL-2003 shared task "language-independent named entity recognition" and shared in two languages: English and German. In this work we will be using the English variant of the dataset, however, everything used can be applied to other languages as the tools used are language independent.

The dataset contemplates four different categories of entities: LOC (Location), MISC (Miscelanious), ORG (Organization) and PER (Person). As annotation follows BIO, each entity is further split between "B" and "I" tags, for example, "B-LOC" notes the beginning of an entity while "I-LOC" would be a word that is part of a multi-word entity, besides this the dataset also has as features POS (Part-of-Speech) Tags and Chunking Tags. It is originally split in 3 sets: Training set, Development set and Testing set. However, due to time constraints, in this work we used only the Training part of the dataset. We used the first 3000 sentences for developing the model and the next 1000 sentences to test their performances. In table 1 we can see more details about the data that was used for both training and testing.

Table 1 Information regarding the data used for training and testing of the models.

Split	Sentences	Tokens	LOC	MISC	ORG	PER
Used for train	3000	54307	1935	986	2255	2659
Used for test	1000	18739	681	447	745	611

2.2 Graph Creation

Besides the features the dataset already has (POS and Chunking tags) we used OntoUD [20] to convert this dataset into a graph and further add more features. This framework converts the text onto a graph and adds features from Universal Dependencies [5] such as the edges, feats, UPOS (Universal POS), XPOS (Optional, language specific POS/Morphological tag) as well as building a dependency graph of each sentence and preserving this structure in the final graph.

Additionally, OntoUD also attempts to match each word to an ID from Wikidata and add it as an attribute of each word as well as create co-reference edges between words using F-Coref [19].

The full sentence is "Germany s representative to the European Union s veterinary committee Werner Zwingmann said on Wednesday consumers should buy sheepmeat from countries other than Britain until the scientific advice was clearer ." and the entities marked are "Germany", " European Union", "Werner Zwingmann", "Britain". In figure 1 we can see this example from the dataset.

4	["Germany", "'s", "representative", "to", "the", "European", "Union", "'s", "veterinary", "committee", "Werner", "Zwingmann", "said", "on", "Wednesday", "consumers", "should", "buy", "sheepmeat", "from", "countries", "other", "than", "Britain", "until", "the", "scientific", "advice", "was", "clearer", "."]	[22, 27, 21, 35, 12, 22, 22, 27, 16, 21, 22, 22, 38, 15, 22, 24, 20, 37, 21, 15, 24, 16, 15, 22, 15, 12, 16, 21, 38, 17, 7]	[11, 11, 12, 13, 11, 12, 12, 11, 12, 12, 12, 12, 21, 13, 11, 12, 21, 22, 11, 13, 11, 1, 13, 11, 17, 11, 12, 12, 21, 1, 0]	[5, 0, 0, 0, 0, 3, 4, 0, 0, 0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0]

Figure 1 Example sentence from the CoNLL dataset [22].

In Figure 2 we can see part of the graph for the above sentence, we can see the edges for a word node, in this case for the node `"word_0_4_13"`. It has a connection to the sentence it belongs (`"Sentence_0_4"`), to the attributes (we can see, for example, that it is a verb in the past tense which is the root of this sentence) which are also represented as edges, and to the words that directly depend on it. Not pictured are the attributes which are: the lemma, NER tags, POS Tags, Chunk Tags and the original word.

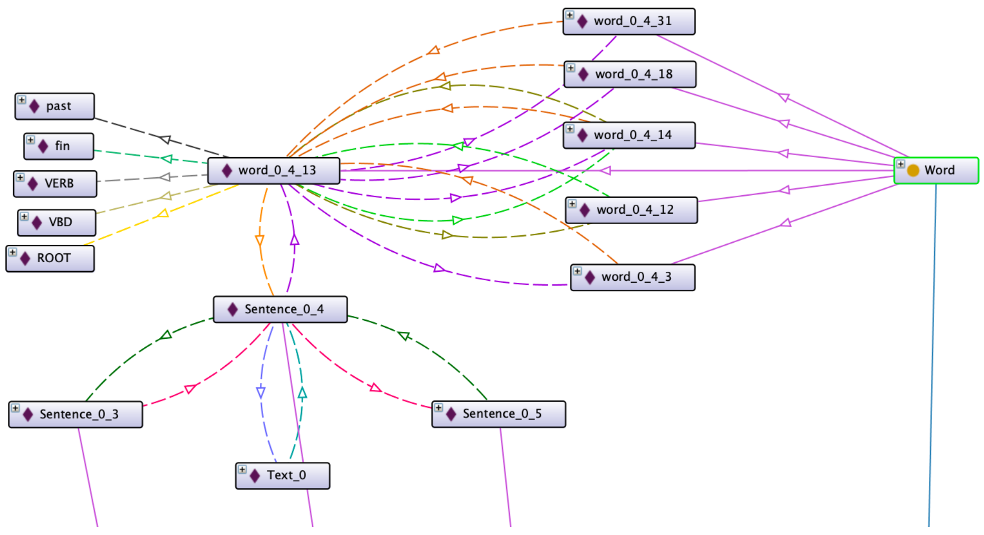


Figure 2 Visualization of the connections for the word node “Word_0_4_13”. Produced using Protégé [17].

This graph is then uploaded onto a triple-storage, in our case, Virtuoso[1] which can then be queried using SPARQL [2].

[1] `https://virtuoso.openlinksw.com/`
[2] `https://www.w3.org/TR/sparql11-query/`

2.3 Embeddings

Since embeddings are large vectors, storing it with the other graph information did not seem the best option. As such, we are generating them before training and testing the algorithms. There was no baseline established for just the use of Embeddings due to our algorithms using Graph ML and Heterogenous Graphs, as such, trying to establish a baseline just for the embeddings would require a completely different model which was not in scope for this work.

To generate these embeddings we adopted Google Cloud[3], more specifically one of their preview models named "text-multilingual-embedding-preview-0409" which can generate multilingual embeddings and provide better results for retrieval and classification [13] since it is what we are trying to achieve by classifying words into a specific NER tag. A list of tokens that make up a sentence is sent to the model and the result is a vector of size 768 for each token. This model was run sentence by sentence as batch predictions are still not supported so it was a time consuming process.

Despite testing on an English dataset we plan on applying this technique to other languages in the future, such as Portuguese and Spanish, making a system that is languange-independent or that requires very minimal changes for it to work.

In order to minimize the time and costs it takes to process the whole dataset we started creating a local database with the words already processed, that consists of a simple dictionary containing pairs of words and their respective vectors. The process starts by querying this dictionary, to see if we already have an embedding for that token or if it is a token that wasn't yet seen. In the end these embeddings make up the final attribute of our Graph before training.

2.3.1 Models

A requirement that we set for this work was that we did not want to work with transductive learning methods and instead opted for an inductive learning approach as these are better at generalizing [11] and do not need access to the testing portion of the graph at training time while for transductive methods need the whole graph at training time. In transductive methods if a new node appears the whole model needs to be re-trained in order to classify it and we want to avoid that.

With this requirement in mind we opted to use a model that uses the operator from GraphSAGE [8] as our main layers. The architecture of the model can be seen in Figure 3. In order to implement and train these models we used PyTorch Geometric [7].

2.3.2 Training

First we had to convert our Graph that was in virtuoso to a Graph Data object before training the model. The conversion process was straightforward. We fetch a sentence and all its dependencies and build the graph that way, however, we had to use a Hetero Data model as not all the nodes in our data have the same connections, for example, the connection for "degree" or "number" does not appear in every node.

Since we are using an Hetero Graph model that means we will have Tensors of different sizes on our network, as such, the initialization of the input channels is not specified by the user but it makes use of lazy initialization by passing the layers the parameter (-1, -1)[4].

[3] `https://cloud.google.com/?hl=en`
[4] `https://pytorch-geometric.readthedocs.io/en/latest/notes/heterogeneous.html#creating-heterogeneous-gnns`

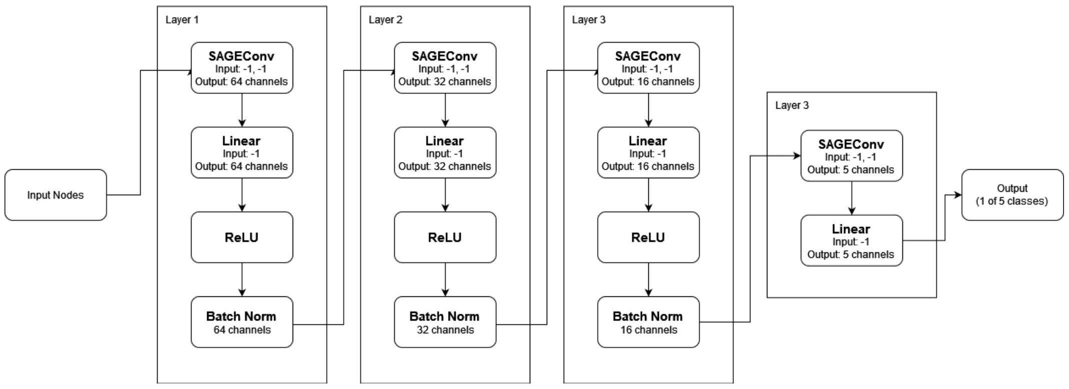

Input Nodes
Layer 1
SAGEConv Input: -1, -1 Output: 64 channels
Linear Input: -1 Output: 64 channels
ReLU
Batch Norm 64 channels
Layer 2
SAGEConv Input: -1, -1 Output: 32 channels
Linear Input: -1 Output: 32 channels
ReLU
Batch Norm 32 channels
Layer 3
SAGEConv Input: -1, -1 Output: 16 channels
Linear Input: -1 Output: 16 channels
ReLU
Batch Norm 16 channels
Layer 3
SAGEConv Input: -1, -1 Output: 5 channels
Linear Input: -1 Output: 5 channels
Output (1 of 5 classes)

Figure 3 Architecture of the model used.

The model then was trained on 3000 sentences from the CoNLL dataset, as mentioned before in Sect 2.1, resulted in 54307 tokens of which 1935 are LOC, 986 MISC, 2255 ORG and 2659 PER and tested on another subset of 1000 sentences with 18739 tokens with 681 LOC, 447 MISC, 745 ORG and 611 PER. The optimizer chosen was Adam [9] with a learning rate of 0.01.

We trained 4 different models, 2 without embeddings and 2 with embeddings. The models differ on their NER tags. For the first set of models we used the original dataset tags, using the “B” and “I” notation. For the second set of models we merged the “B” and “I” into a single class, for example, “B-PER” and “I-PER” would just become “PER”.

3 Results

The results for the 4 models in four common metrics in the NER field are presented in Fig. 4.

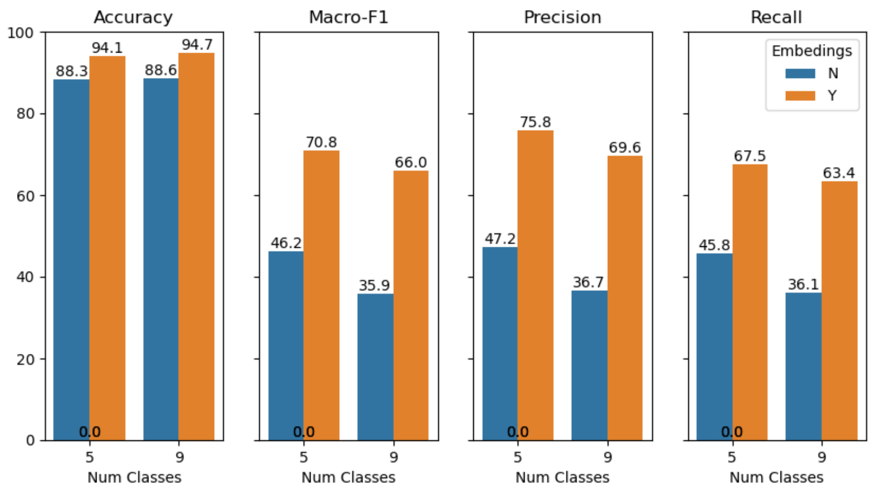
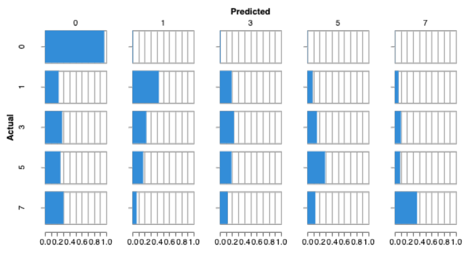
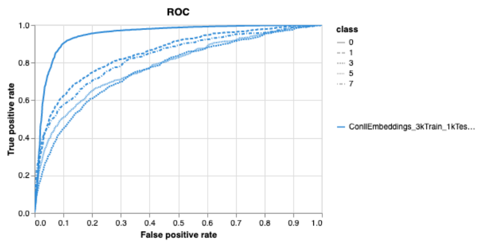


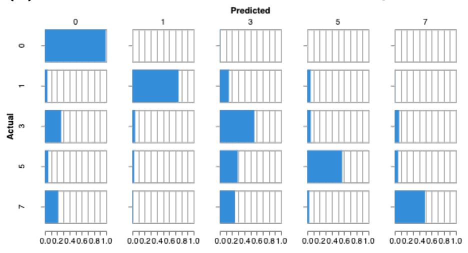
Figure 4 Effect of Embedings and number of classes in NER performance. Four common metrics are shown: Accuracy, Macro F1, Precision and Recall.



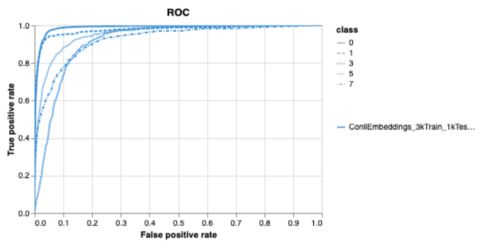
(a) Confusion Matrix, No Embeddings



(b) ROC Curve, No Embeddings



(c) Confusion Matrix, Embeddings



(d) ROC Curve, Embeddings

Figure 5 Confusion Matrix and ROC Curves for the models with 5 classes with and without embeddings.

The results in the figure show a positive effect of embeddings, for both number of classes. The results with the higher number of classes are aprox. 10% lower in terms of Precision, Recall and Macro-F1. This difference between 5 and 8 classes could be explained by, the 8 classes needing to have more training examples to achieve better results or by there not being a lot of difference between, for example, a I-LOC and an I-ORG.

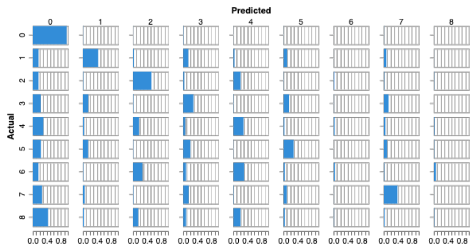
While the accuracy is not that different, due to imbalance of the dataset, this metric can be misleading. However, looking at the Macro-F1, Precision and Recall there is a clear improvement. By using Embeddings the results were boosted significantly. When looking at the accuracy of the models there is only a $\approx 6\%$ difference between the models, however, looking at the other metrics it is clear that adding embeddings information as an attribute of each word drastically improves the results with the Macro F1, Precision and Recall having an improvement of over 20%.

There are obviously some implications by removing the BIO tags from the data. However, if we can properly detect when an entity begins our hypothesis is that by using the dependency graph of the beggining of this entity we can extract the full entity. This hypothesis was not tested in this document and will be looked at in future work.

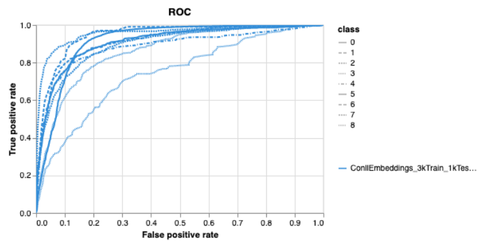
To better evaluate the performance of the models next we will go in deeper detail, supported by confusion matrices and ROC curves, starting by 5 classes.

3.1 Results with 5 classes

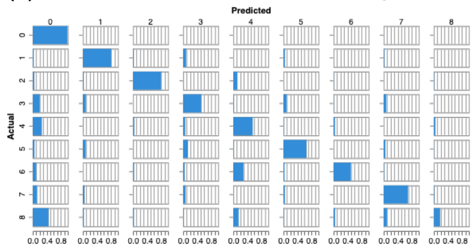
Starting off talking about the model without embeddings, we can see on the normalized confusion matrix 5a that this model, unsurprisingly, only had a good performance detecting when there was no entity present with the best performance being on prediction on the class labeled as 1 (PER) with it being 43.21%. From the ROC Curve 5b we can see that the model is better than a random classifier, however, it is far from being a good classifier. This fact is also reflected on the metrics we were looking at. The macro f1-score, precision and recall for this model were 46.23%, 47.16% and 45.77% respectively.



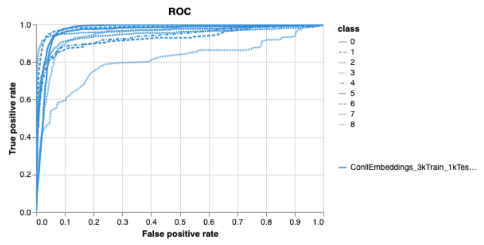
(a) Confusion Matrix, No Embeddings



(b) ROC Curve, No Embeddings



(c) Confusion Matrix, Embeddings



(d) ROC Curve, Embeddings

Figure 6 Confusion Matrix and ROC Curves for the models with 9 classes with and without embeddings.

Now looking at these same metrics for the embedding versions we can see normalized confusion matrix (Figure 5c) and the ROC Curve (Figure 5d) show improvement on both metrics. On the confusion matrix the best result for an entity jumped from 43.21% to 75.45% for the same label (1 - PER) while for the worst label (3 - ORG) went from 23.22% to 56.38%. The ROC Curve also shows clear improvements with the curves for the classes being closer to the ideal one. This is, once again, backed up by the results of the metrics with them increasing to 70.81%, 75.79% and 67.48% for Macro-F1, Precision and Recall.

3.2 Results with 9 classes

The results for the model without embeddings and with even more classes were, as expected, worse than the previous ones. While the accuracy of the model was comparable to the one with 5 classes, 88.51%, the f1-score, precision and recall were all worse with this model achieving 35.86%, 36.74% and 36.07% respectively. These values show a decrease in performance of about 10%.

In figures 6a and 6b the results are similar to the 5 classes model without embeddings, and the best label the model could predict was "2 - I-PER". The ROC curve is also far from being ideal with the worst performing class being "8 - I-MISC", which is not surprising due to the low amount of data this label has.

When comparing this model to the previous one, once again, the similarities to the 5 classes model are there. There is a drastic improvement in results. The accuracy saw an improvement from 88.51% to 94.73%, however, the biggest jump is, once again, in F1-score, precision and recall with this model having 65.98%, 69.62% and 63.40%, a jump of over 30% when compared to the model with no embeddings. The confusion matrix shows that the best class was 1 ("B-PER") with 81.16% correct predictions and the worst one 8 ("I-MISC") with 17.8%. The ROC Curve shown in Figure 6d also shows a drastic improvement over the previous one, except for class 8 as it does not have a lot of data points for the algorithm to predict on.

4 Conclusion and Future Work

In this paper we assessed the use of embeddings in graph-based inductive ML when applied to NER. By using graph-based methods we are able to keep the syntactic structure of the text intact as well as any connections between words and combining it with embeddings we get contextual information improving our models this way. The use of inductive Graph ML algorithms, in this case, Graph SAGE [8] also makes the use of these models more applicable to real-world scenarios as whenever new documents show up the model can handle them.

This results of this work show that adding embeddings for NER improves the results even when using Graph Machine Learning. The combination of having a graph that represents syntactic dependencies between words with the contextual information that embeddings can achieve boosts the capabilities of the algorithms even in a very imbalanced setting like NER.

Since we are using graphs, a further study could be done on the 5 classes model by looking at the sub-graphs that can be constructed from a word that is classified as an entity. For example, a word classified as the beginning of the entity we could check the sub-graph of that word (based on the dependency graph) to see if it is a multi-word entity or a single-word entity.

Besides this, for future work we would like to test different architectures and configurations and apply this work to other languages and domains to see if the results stay consistent. Furthermore, applying this approach to other problems, for example, Open Information Extraction should also be possible without changing a lot.

References

1 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. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

2 Manuel Carbonell, Pau Riba, Mauricio Villegas, Alicia Fornés, and Josep Lladós. 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, 2021. `doi:10.1109/ICPR48806.2021.9412669`.

3 Alberto Cetoli, Stefano Bragaglia, Andrew O'Harney, and Marc Sloan. Graph convolutional networks for named entity recognition. In Jan Hajič, editor, *Proceedings of the 16th International Workshop on Treebanks and Linguistic Theories*, pages 37–45, Prague, Czech Republic, 2017. URL: `https://aclanthology.org/W17-7607`.

4 Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and Jiliang Tang. Exploring the potential of large language models (llms)in learning on graphs. *SIGKDD Explor.*, 25(2):42–61, March 2024. `doi:10.1145/3655103.3655110`.

5 Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 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, May 2014. European Language Resources Association (ELRA). URL: `http://www.lrec-conf.org/proceedings/lrec2014/pdf/1062_Paper.pdf`.

6 Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019. URL: `https://api.semanticscholar.org/CorpusID:52967399`.

7 Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

8 William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018. `arXiv:1706.02216`.

9 Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. `arXiv:1412.6980`.

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

11 Guillaume Lachaud, Patricia Conde-Cespedes, and Maria Trocan. Comparison between inductive and transductive learning in a real citation network using graph neural networks. In *Proceedings of the 2022 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '22, pages 534–540. IEEE Press, 2023. `doi:10.1109/ASONAM55673.2022.10068589`.

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

13 Jinhyuk Lee, Zhuyun Dai, Xiaoqi Ren, Blair Chen, Daniel Cer, Jeremy R. Cole, Kai Hui, Michael Boratko, Rajvi Kapadia, Wen Ding, Yi Luan, Sai Meher Karthik Duddu, Gustavo Hernandez Abrego, Weiqiang Shi, Nithi Gupta, Aditya Kusupati, Prateek Jain, Siddhartha Reddy Jonnalagadda, Ming-Wei Chang, and Iftekhar Naim. Gecko: Versatile text embeddings distilled from large language models, 2024. `arXiv:2403.20327`, `doi:10.48550/arXiv.2403.20327`.

14 Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. `doi:10.18653/v1/2020.acl-main.703`.

15 Monica Madan, Ashima Rani, and Neha Bhateja. Applications of named entity recognition using graph convolution network. *SN Computer Science*, 4(3):266, 2023. `doi:10.1007/S42979-023-01739-8`.

16 Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013. URL: `https://api.semanticscholar.org/CorpusID:5959482`.

17 Mark A. Musen. The protégé project: a look back and a look forward. *AI Matters*, 1(4):4–12, 2015. `doi:10.1145/2757001.2757003`.

18 R OpenAI. Gpt-4 technical report. *ArXiv*, 2303, 2023.

19 Shon Otmazgin, Arie Cattan, and Yoav Goldberg. F-coref: Fast, accurate and easy to use coreference resolution, 2022. `arXiv:2209.04280`, `doi:10.48550/arXiv.2209.04280`.

20 Gabriel Silva, Mário Rodrigues, António Teixeira, and Marlene Amorim. A Framework for Fostering Easier Access to Enriched Textual Information. In Alberto Simões, Mario Marcelo Berón, and Filipe Portela, editors, *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, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/OASIcs.SLATE.2023.2`.

21 Gabriel Silva, Mário Rodrigues, António Teixeira, and Marlene Amorim. First assessment of graph machine learning approaches to Portuguese named entity recognition. In Pablo Gamallo, Daniela Claro, António Teixeira, Livy Real, Marcos Garcia, Hugo Gonçalo Oliveira, and Raquel Amaro, editors, *Proceedings of the 16th International Conference on Computational Processing of Portuguese*, pages 563–567, Santiago de Compostela, Galicia/Spain, March 2024. Association for Computational Lingustics. URL: `https://aclanthology.org/2024.propor-1.61`.

22 Erik F. Tjong Kim Sang and Fien De Meulder. 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, pages 142–147, 2003. URL: `https://aclanthology.org/W03-0419`.

23 Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings, 2016. `arXiv:1603.08861`.