Multi-Agent Systems

Project Report

Collaborating NLP Agents

Leonidas Gee
leonidas.gee@studio.unibo.it

Miki Mizutani
miki.mizutani@studio.unibo.it

Academic Year: 2021 / 2022

1 Abstract

The project involves the design and implementation of a multi-agent system for training classifiers with different configurations given a parameter space. Using a master-slave architecture, the system searches the parameter space by random sampling and trains the classifiers for named-entity recognition on the CoNLL-2003 dataset in a round-robin manner. The best trained classifier is then determined and evaluated on the test set. Results show that the proposed multi-agent system is able to search the parameter space efficiently.

2 Introduction

In the field of natural language processing (NLP), the training and validation of language models can be a tedious process. The choice of feature, model architecture, and number of neural units can have a significant effect on the final performance of a language model. As such, it is typical for practitioners to train multiple models using different configurations in order to determine the best possible model. Such a process is usually time consuming and dependent on the size of the parameter space to be searched.

To overcome this issue, a multi-agent system (MAS) may be used to efficiently search the parameter space when training the language models. Conceptually, a given number of agents may be initialised as classifiers with randomly sampled configurations. These agents may be trained in a round-robin manner or in parallel using a cluster of computers. A possible design for such a system would be a master-slave architecture, whereby a controller (master) would assign randomly sampled configurations to a group of classifiers (slaves) for training and validation.

To test such a proposed system, an named-entity recognition (NER) task based on the CoNLL-2003 dataset [6] was selected. The goal of the project would then be to design a MAS able to satisfy the technical requirements of the proof of concept as defined above.

The remainder of the report is organised as follows:

- **Section 2 - Proposal**: describes the project proposal
- **Section 3 - Dataset**: describes the dataset used
- **Section 4 - Method**: describes the methodology developed
- **Section 5 - Results**: discusses the results
- **Section 6 - Conclusion**: draws the conclusions

3 Proposal

Our proposal is based on a project proposal at the University of Edinburgh titled: Collaborating NLP Classifiers as Agents in a Multi-Agent-System [4]. The original proposal envisions the development of the basic processes in creating and teaching a NLP classifier by means of communicating agents in a Multi-Agent environment. For a start, this will include selecting features as well as training, tweaking and testing on data.

Our modification of the said proposal involves the inclusion of only feature selection, training, and tweaking in the MAS to be developed. On the other hand, the testing phase of the NLP pipeline is kept separate from the MAS in order to quantitatively and qualitatively analyse each classifier's performance separately. Our proposed modified methodology will serve as a proof of concept in merging the technologies of MAS and NLP together.

3.1 Task

The main task that the system is designed to solve is one based on named-entity recognition (NER). The main goal of NER is to identify predefined semantic types such as persons, locations, organisations, etc. from a given text. The benefits of NER stem not only from its use in information extraction, but also its usefulness as a feature for downstream NLP tasks such as question answering, machine translation, etc. In general, the named-entities (NEs) can be divided into two types: generic NEs (e.g., person and location) and domain-specific NEs (e.g., proteins, enzymes, and genes) [2]. For this project, the NEs to be recognised in the dataset are generic ones.

4 Dataset

The shared task of CoNLL-2003 concerns language-independent named-entity recognition. It focuses on four types of named entities, namely: persons, locations, organisations and names of miscellaneous entities that do not belong to the previous three groups. The dataset has already been divided into a training set, a validation set, and a test set.

The CoNLL-2003 dataset contains four columns separated by a single space. Each word has been put on a separate line and there is an empty line after each sentence. The first item on each line is a word, the second a part-of-speech (POS) tag, the third a syntactic chunk tag, and the fourth the named-entity tag. All three features have tags that numerically start from 0.

4.1 Preprocessing

In general, the dataset has already been heavily preprocessed with three different features provided for modelling. However, as we intend to subsequently use neural-based language models, zero padding will be required to properly train such models. As such, each of the three provided features have had their tags numerically incremented by 1 such that 0 is reserved for the eventual zero padding tags.

4.2 GloVe Embeddings

In addition to the three features provided by the CoNLL-2003 dataset, a fourth feature is also added that involves the embedding of the tokens using GloVe vectors [3]. Word embeddings are, in simple terms, a way of quantitatively representing the meaning of a token using a vector of continuous values. The higher the dimension of such vectors, the more representative is the meaning of the token that is encoded.

Ever since the invention of GloVe embeddings in 2014, newer research has produced better embeddings such as FLAIR embeddings that are able to take context into account [1]. However, due to RAM limitations, we decided to use smaller dimensional embeddings such as the 50-dimensional GloVe embeddings to encode our tokens.

To achieve this, the tokens are first lowercased before being embedded as a 50-dimensional vector using the glove-wiki-gigaword-50 vectors. These vectors have been pretrained using the Wikipedia 2014 and Gigaword 5 datasets. Furthermore, in order to handle out-of-vocabulary (OOV) words, which are words that do not have an associated GloVe vector, a random 50-dimensional vector is assigned instead. Care is taken to ensure that each OOV word is only associated with a single random vector and that two of the same OOV words will not have different random vectors.

5 Method

5.1 Multi-Agent System

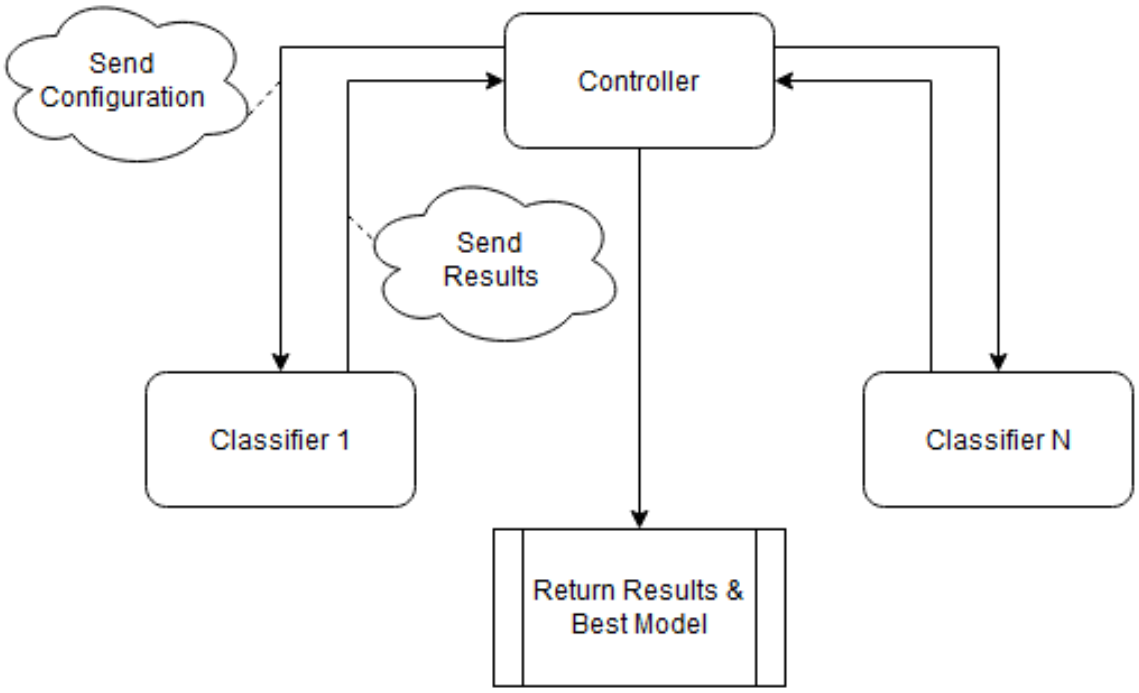

Figure 1: Architecture of the proposed multi-agent system. There is a single controller with N number of initialised classifiers. Communication between the controller and classifiers is two-way.

To model the MAS, the osBrain library is used. The library is a general-purpose multi-agent system module written in Python and developed by OpenSistemas. The library was chosen as it allows for easy integration with existing NLP technologies in Python such as Keras.

The MAS proposed is based on a master-slave architecture as shown in Figure 1. In such an architecture, the master represents the single controller, while the slaves represent the K number of initialised classifiers. Due to RAM limitations, the number of classifiers to be initialised has been set to 2.

5.2 Modelling

To model the dataset, a recurrent neural network (RNN) is used for each classifier. The architecture of the RNN depends on the parameters given to it. The following is the general design of the RNN:

- Input Layer: accepts features of dimension 50 if GloVe embeddings are used or accepts features of dimension 1 if pos tags or chunk tags are used.
- BiLSTM / BiGRU Layer: a bidirectional LSTM or GRU layer with a variable number of units dependent on the configuration used.
- Dense Layer: a time distributed dense layer with 10 units and a final softmax function for multiclass classification.

Finally, each RNN uses a categorical crossentropy loss function and an Adam optimiser. Early stopping based on the validation loss of the models is also implemented. To reduce the amount of RAM used by each classifier, generators are initialised to ensure that data is streamed with a batch size of 8 for training and validation. The accuracy metric used by the RNN is a custom metric that is designed to ignore the zero padding in the predictions.

5.3 Parameter Space

The parameter space is all the possible combinations of parameters that can be used to define the RNNs. For this project, three parameters are used, namely: model type, feature type and number of neural units. The defined parameters have the following values:

- Model Type: LSTM, GRU
- Feature Type: GloVe embeddings, POS tags, Chunk tags
- Number of Neural Units: a distribution between 32 and 128 with a step of 32

5.4 Training

During training, the name server is first initialised by the system. This is followed by the initialisation of the controller and the K number of classifiers. To configure the system, two types of addresses based on the PUSH-PULL communication are defined for the agents. As the PUSH-PULL communication is unidirectional, using two types of addresses allows for bidirectional communication between the agents.

After connecting the agents to their respective addresses, the controller is given the lists of possible values for each parameter, the number of epochs, the batch size, and the random seed for training and validation. The controller will then determine all the possible combinations of parameters that can be used as configurations by the classifiers.

A parameter N is also defined that represents the number of times to run the parameter search and model training by the system. For each iteration of N, the controller will first send a randomly sampled combination of parameters from the parameter space as a message to each classifier. Each classifier will then set up their RNN using the parameters received and start training the models accordingly. The classifiers are all trained in a round-robin manner.

Once a classifier has been trained and subsequently validated on the validation set, its model configuration, validation loss, and validation accuracy are sent to the controller for storage in a common results dictionary. The classifier will also save the trained model to disk.

This process of random sampling and training is repeated for each of the N iterations. Once the N iterations are complete, the controller returns the results and the best model configuration by sorting the classifiers in descending order of validation accuracy. The system is then finally stopped by shutting down the name server.

6 Results

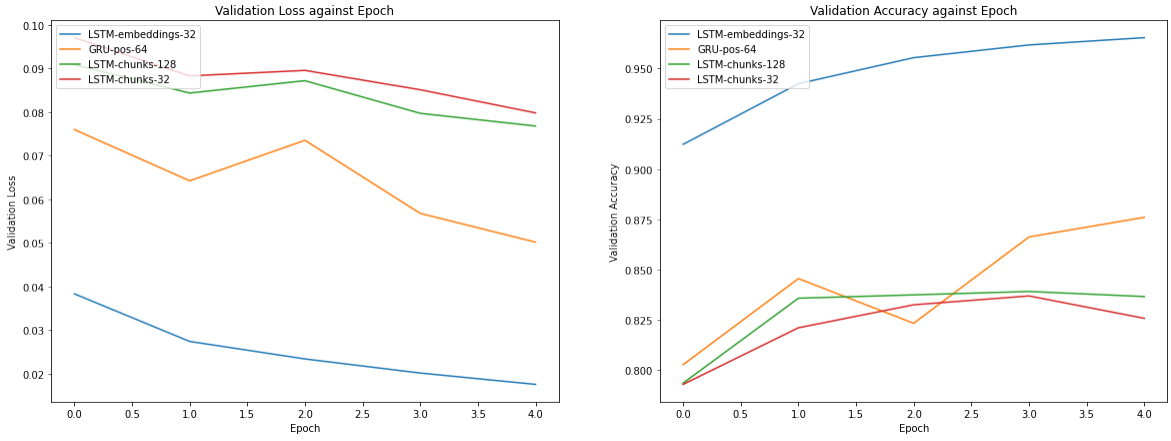

Figure 2: Validation loss and accuracy of the trained classifiers.

Figure 2 shows the 4 models sampled and trained by the MAS using 5 epochs. We can see that among all the parameter types, the feature type parameter has the greatest influence on the final validation loss and accuracy. In particular, the usage of GloVe embeddings seems to provide a larger gain in performance compared to the POS tags and chunk tags. This is most likely due to the larger

dimension of the GloVe embeddings, 50 in this case, that provides more encoded information to the RNNs in solving the NER task.

Furthermore, we may observe that the number of neural units of the bidirectional layer seems to have the least influence on the final validation loss and accuracy of the models as shown by the relatively similar performance between the LSTM-chunks-128 and LSTM-chunks-32 models.

	Precision	Recall	F1-Score	Support
PAD	0.00	0.00	0.00	0
O	0.98	0.99	0.98	38316
B-PER	0.88	0.91	0.89	1616
I-PER	0.93	0.95	0.94	1155
B-ORG	0.80	0.75	0.77	1660
I-ORG	0.79	0.61	0.69	835
B-LOC	0.88	0.86	0.87	1667
I-LOC	0.74	0.69	0.71	257
B-MISC	0.72	0.70	0.71	702
I-MISC	0.60	0.57	0.58	216
accuracy			0.95	46424
macro avg	0.73	0.70	0.71	46424
weighted avg	0.95	0.95	0.95	46424

Table 1: Classification report of the best model (LSTM-embeddings-32) on the test set.

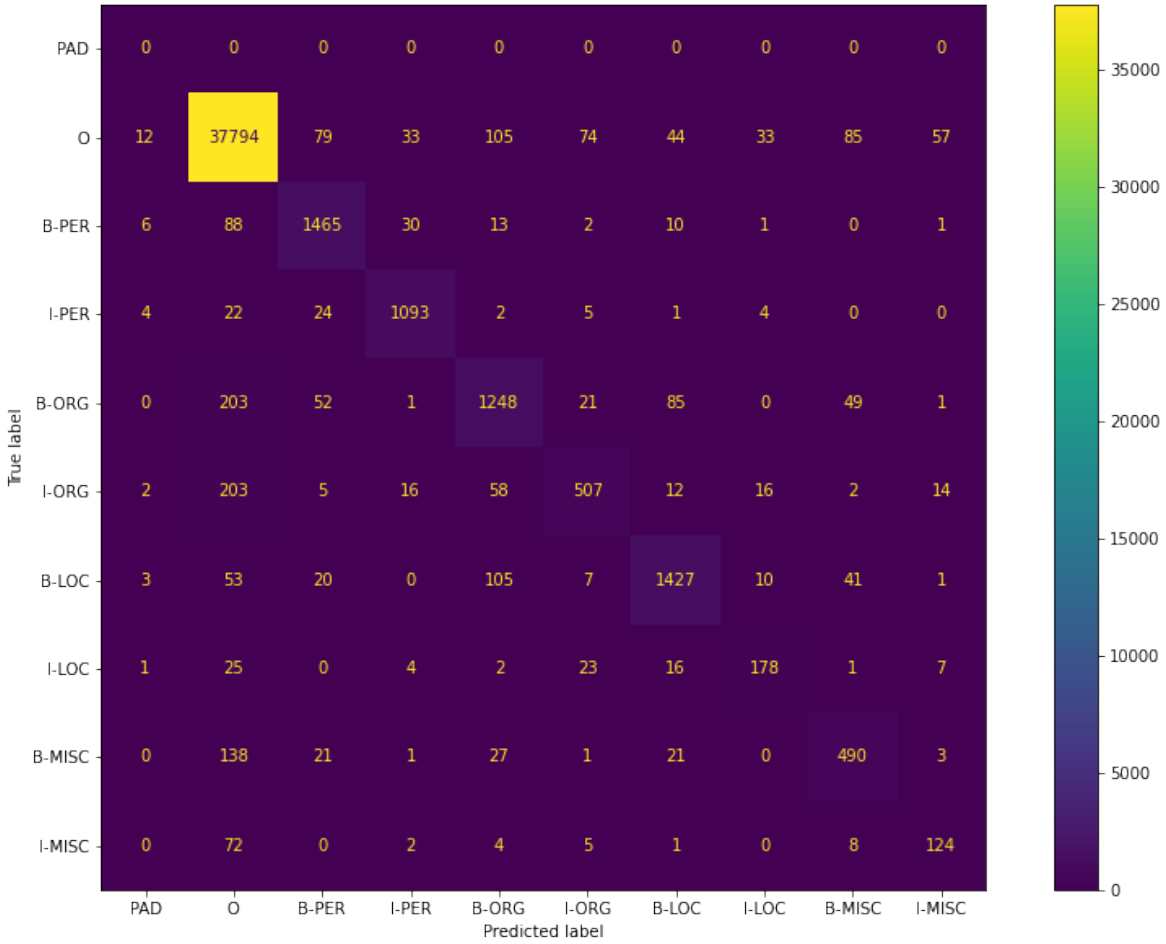

Figure 3: Confusion matrix of the best model (LSTM-embeddings-32) on the test set.

Using the validation accuracy shown in Figure 2, we select the best model to be evaluated on the

test set. In this case, the best model corresponds to the LSTM-embeddings-32 configuration.

Based on our achieved results in Table 1, we note certain findings. Firstly, our best model produces a high accuracy score of 0.95, but a much lower accuracy score if the macro average is applied (when classes are weighted equally regardless of sample size). This implies that the model does not perform equally well in the classification of all the possible classes. Certain classes such as the O tag are classified very well by the model, but others such as the I-MISC tag have a poorer performance. We can also observe from the confusion matrix in Figure 3 that the model still misclassifies certain tags as zero padding.

Finally, we acknowledge how the number of iterations and the number of initialised agents will affect the final choice of the best model. Hypothetically, if the MAS system is allowed to fully explore the parameter space by sampling all possible configurations, the best possible model may be found. This can be done by running more iterations if there is a single computer or by initialising multiple agents on separate computers in a cluster of computers. However, due to the technical constraints of the available hardware, the best model achieved in this project is done by approximation using a few iterations and agents.

6.1 State of the Art

The current state of the art (SOTA) approaches in solving the English NER task of the CoNLL-2003 dataset far exceeds the achieved F1-score of the best model found by our system. This in part can be attributed to the far better approaches in terms of word embeddings and language models used.

In recent years, the field of NLP has moved on to the use of contextual embeddings and transformers for NER. These approaches have been shown empirically to produce far better results in downstream language modelling tasks. For example, the current SOTA approach for our given problem by [7] utilises neural architecture search to find the best combination of contextual embeddings for solving the NER task. In addition, another top approach by [5] utilises transformer-based models for capturing document-level features in order to solve the same given task.

In theory, if hardware limitations were not a factor, it would be possible to use our current MAS system as a framework for solving the NER task with more complex NLP approaches such as transformers. However, due to RAM limitations, we decided to use more traditional approaches in realising the proof of concept for our proposed MAS system.

Despite the lower F-1 scores of this project, using a MAS-based approach brings the benefit of saved time, efficient testing of other architectures, and easy comparison of results. For the future, it could be possible to have communication between agents, where if a parameter gives bad results (such as a low learning rate), then other agents will take a different approach (such as train with a higher learning rate).

One limitation of a MAS approach to parameter selection is the inability to cancel a training midway. If implemented on a model such as a transformer, with long training times, it may be necessary to implement criteria for early abandoning.

7 Conclusion

Given our achieved results, we have realised a proof of concept that merges the technologies of MAS and NLP. Our project shows how a MAS can be used to train RNNs using collaborating agents in the form of a master-slave architecture. At the moment, the developed MAS trains the initialised classifiers in a round-robin manner. However, if a cluster of computers is available, each classifier can be trained separately on a single computer, thus significantly speeding up the exploration of a parameter space in an NLP pipeline. This framework can possibly be applied to other domains such as computer vision.

References

[1] Alan Akbik, Duncan Blythe, and Roland Vollgraf. "Contextual String Embeddings for Sequence Labeling". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 1638–1649. URL: https://aclanthology.org/C18-1139.

[2] Jing Li et al. *A Survey on Deep Learning for Named Entity Recognition*. 2018. DOI: 10.48550/ARXIV.1812.09449. URL: https://arxiv.org/abs/1812.09449.

[3] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global Vectors for Word Representation". In: vol. 14. Jan. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162.

[4] Sebastian Riedel. *Collaborating NLP Classifiers as Agents in a Multi-Agent-System*. Sept. 2004. URL: http://www.inf.ed.ac.uk/teaching/courses/diss/03-04/props/255_riedel1.html.

[5] Stefan Schweter and Alan Akbik. "FLERT: Document-Level Features for Named Entity Recognition". In: *CoRR* abs/2011.06993 (2020). arXiv: 2011.06993. URL: https://arxiv.org/abs/2011.06993.

[6] 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*. 2003, pp. 142–147. URL: https://aclanthology.org/W03-0419.

[7] Xinyu Wang et al. "Automated Concatenation of Embeddings for Structured Prediction". In: *CoRR* abs/2010.05006 (2020). arXiv: 2010.05006. URL: https://arxiv.org/abs/2010.05006.