Domain-Adaptive Learning with Gradient Reversal and Wordle-Inspired Training for Named Entity Recognition

Alessio Cappello
Politecnico di Torino
Turin, Italy
s309450@studenti.polito.it

Giulia Di Fede
Politecnico di Torino
Turin, Italy
s307746@studenti.polito.it

Alberto Foresti
Politecnico di Torino
Turin, Italy
s309212@studenti.polito.it

Claudio Macaluso
Politecnico di Torino
Turin, Italy
s317149@studenti.polito.it

***Abstract*—In the legal domain, Legal Named Entity Recognition (L-NER) has become an established Natural Language Processing (NLP) technique able to aid legal experts, such as attorneys and judges in domain-specific activities like content retrieval and decision-making. Since incorrect recognition can lead to incorrect legal decisions, the high accuracy of NER is crucial for the safe analysis of legal documents. In this paper, we extend a previous work on L-NER by proposing (1) a new training task based on the game 'Wordle' and (2) we aim to experiment on domain adaptation exploiting a gradient reversal layer to close the domain shift between legal data and defence data. With this approach, we can investigate the use of a domain classifier both at the token level and at the window level, understanding which performs best. We designed a new architecture composed of an entity classifier, a game module to handle the hints' mechanism and domain classifiers at both token and window level. The architecture takes as input the contextualised embeddings provided by a transformer-based model, in our case BERT, and can work by exploiting a multi-task learning framework to evaluate the effectiveness of the approach. Even though our proposed extensions did not accomplish state-of-the-art performance, we gained useful insights about the two domains and the BERT contextualised embeddings, that can be considered for future work in this field to gain remarkable performance.**

***Index Terms*—Named Entity Recognition, Gradient Reversal, Domain Adaptation, Multi-Task Learning**

I. Introduction

In recent years, Natural Language Processing tools have demonstrated great agility in aiding a wide range of applications in different domains. Pre-trained language models currently represent state-of-the-art architectures in several downstream tasks. However, while most of the literature focuses on less specific domains, little exploration has been done in the legal domain.

Like other specialised domains, the legal domain is delineated by its specific and complex vocabulary, a variety of scenarios and the temporal evolution of regulations: expedients have to be taken to achieve acceptable performance.

The *Named Entity Recognition* (NER) task applied in a legal context is commonly referred to as *Legal-NER* (L-NER). Transformer-based models can be exploited to address the task, and it has already been proved that they can obtain remarkable performance: however, further strategies can be explored to push the limit.

Another common technique in deep learning is domain adaptation: it consists of training or fine-tuning a model on a source dataset, and testing it on a target dataset. Some measures have to be embraced since the feature distributions of the two datasets may differ significantly. An affirmed strategy is the one proposed in (1), where an unsupervised adversarial framework is embraced and the objective is to fool a domain classifier while the two domains' representations are brought closer.

It's also possible to rely on Multi-Task Learning (2) where multiple tasks are solved simultaneously during the training stage, all based on the same shared representation.

In this paper, we propose:

- a domain adaptation algorithm based on the gradient reversal layer strategy;
- a new training task inspired by the game 'Wordle', where a hint is given based on the token position and the entity type.

The source code can be found here.

II. Related Works

A. Named Entity Recognition

Named Entity Recognition is a task that aims to locate in some text and classify named entities, that refer to the real world and are typically associated with a proper name, such as a person, organisation, location, date or other. One of the different approaches to performing NER is the so-called transformer-based NER, which uses a transformer architecture to produce word embeddings to apply NER. Different pre-trained transformer architectures are available and can be used to perform this task: one of the most established ones is BERT, pre-trained on a large corpus of text from various sources. Besides its state-of-the-art results on general-purpose NER tasks, BERT has demonstrated suboptimal performance when applied to specialised domains: in fact, certain domains use such a specific language that can bring up some challenges for general-purpose models. To address these issues, in (3) was proposed to take BERT and try to specialise the model, using two different pre-training strategies:

- further training on legal corpora;
- performing the whole pre-training on the legal corpora.

This work has shown that, for specific domains, specialising the model is a good choice and gives better results than BERT's general-purpose version.

B. *Multi-Task Learning*

Common challenges with the traditional training approach are the need for large amounts of data and the computational demand. To mitigate these worries, it is possible to let a model solve different tasks at the same time: this way, the model learns a shared representation that also has a regularisation effect. What has been said falls under the term Multi-Task Learning (2). This framework aims to make a model learn a representation that can be used to address other downstream tasks faster, preferring a more general knowledge instead of a specific vision for a single task. Each task is carried out independently trying to optimise their losses, but care has to be taken to limit the negative transfer phenomenon, namely when an increase in performance for one task leads to having worse performance for another.

C. *Domain Adaptation*

Domain adaptation is a diffused transfer learning technique used to leverage the knowledge acquired from a specific domain and apply it to another one. Among several existing approaches to perform domain adaptation, a strategy that made its way is Adversarial Domain Adaptation, in which we have:

- *Feature Extractor* is a network that tries to learn a common representation for data belonging to the two domains;
- *Domain Discriminator* is trained to distinguish between the two domains.

The discriminator is connected to the extractor through a Gradient Reversal Layer (1), used since the two parts have contrasting objectives: the feature extractor aims to learn an indistinguishable representation. In contrast, the domain discriminator aims to recognise to which domain a representation belongs. This approach can be employed in supervised and unsupervised settings, with different domains potentially having disjoint target classes. In our work, we experiment with two domains where classes slightly overlap.

III. System Overview

A. *Problem statement*

The L-NER task (4) involves identifying and classifying specific legal entities within unstructured legal texts. It is possible to exploit various training strategies to accomplish this task: we took inspiration from the game Wordle, where the goal is to guess a word and hints are given based on the presence/absence of the characters and their position. We also analysed the performance of an adversarial domain adaptation approach to evaluate the performance on a different domain, more specifically the defence one.

B. *Methodology*

We propose two extensions to extend the work conducted in (5):

- a task that can be used in the training stage, based on the game 'Wordle', giving hints to guide the learning process;
- an adversarial domain adaptation approach to explore the possibilities of exploiting knowledge acquired on a certain domain to perform NER on another one.

A diagram of the whole architecture is shown in fig. 1. We used BERT-large fine-tuned on NER. We used the pre-trained checkpoint available on the Hugging Face hub repository[1] to fine-tune it on the Indian Legal Document Corpus (ILDC) (6) to extract the word embeddings, adopting sum pooling over the last four hidden layers. The collected outcome is given as input to our designed model: since it is meant to be general, it has some repeated components for the source and target domain. All rely on the Task Module, which is a Fully Connected network with one fully connected layer, and it is the basis for the following components:

- *Classifier*, that outputs the predictions in terms of entities;
- *Token Domain Classifier*, that aims to determine to which domain a token belongs and is composed of two fully connected layers;
- *Window Feature Extractor*, that extracts the features at the window level.

The features extracted by the last component are used for these two modules:

- *Window Domain Classifier*, same as the token one but refers to a window;
- *Game Module*, that manages the Wordle-inspired system and is composed of a fully connected layer.

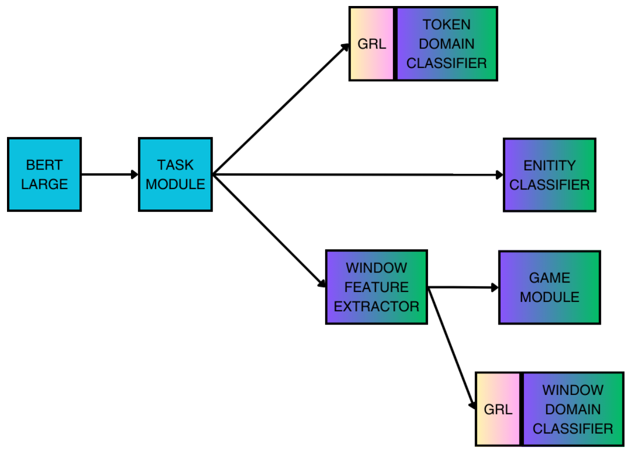

Fig. 1. Diagram of the proposed architecture

The game module, in particular, concatenates the features of the window with a hint and the last attempt at the game and passes them to a fully connected layer to calculate the new attempt at the game. The objective of the game is to guess the entities in the window in the correct position. The last attempt

[1]BERT-large repository: https://huggingface.co/dslim/bert-large-NER

is computed by taking, for every position in the window, the entity that the module is more confident in guessing. The hint vector is composed by setting each entry guessed correctly to 1, while wrong guesses are set to 0. Then the hint and the last attempt are used to make another guess with the window features, this process is repeated a number of times n, which is a hyperparameter. A diagram of this module is illustrated in fig. 2.

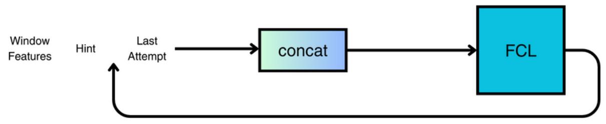

Fig. 2. Diagram of the wordle module

For the domain adaptation study, we considered the ILDC dataset as the source one and the R3ad dataset (7) as the target one, which is a collection of entities and relationships related to the defence domain. Specifically, we focused on identifying and finding the following entities: identifiers for relevant elements, identifiers of documents, radio frequencies, locations, military platforms, money, nationalities, organisations, people, quantities, temporal references, URL, vehicles and weapons.

1) Loss functions: In our multi-task setting, we designed several loss functions to accommodate different learning objectives. We used a cross-entropy loss $\mathcal{L}_{clf}$ for entity classification and for both token domain classification $\mathcal{L}_d^{(t)}$ and window domain classification $\mathcal{L}_d^{(w)}$. Whereas, for what concerns the game module, we defined two losses based on cross-entropy loss that take into account the logits of the last attempt of the network at the game:

- *Wordle position loss* $\mathcal{L}_{wordle}^{(p)}$: we apply a cross-entropy loss for every token in the window to understand how the network was able to detect the entity at the right place.
- *Wordle window loss* $\mathcal{L}_{wordle}^{(w)}$: from the logits of the last attempt we can assess the probability that the model predicts that some kind of entity will be in the window. More formally, given a window w, with length N:

$$p_e = 1 - \prod_{i=1}^{N}(1 - P(w_i = e))$$

Then, if we have E entities, we calculate $\mathcal{L}_{wordle}$ as:

$$\mathcal{L}_{wordle}^{(w)} = -\frac{1}{E}\sum_{k=1}^{E} n_k \hat{p}_k \log p_k$$

where $\hat{p}_k$ is the ground truth of whether the k-th entity is in the window or not and n_k is the number of times the k-th entity appears in the window.

For every loss we use, we perform the computation for both source and target and sum the losses for source and target to get our final loss. In the end, the final loss is:

$$\mathcal{L} = \beta_{\text{wordle}}(\mathcal{L}_{wordle}^{(p)} + \mathcal{L}_{wordle}^{(w)}) + \beta_{\text{token}}\mathcal{L}_d^{(t)} + \beta_{\text{window}}\mathcal{L}_d^{(w)} + \mathcal{L}_{\text{clf}}$$

where the beta parameters are used to balance the weights of the different losses.

IV. Experimental Results

In this section, we report our work settings, the used evaluation metrics and the results of our investigation of the problem.

A. Experimental Design

1) Hardware: Experiments were run using a machine equipped with Intel® Xeon® Gold 6342 CPU and NVIDIA® A100 80GB GPU running Ubuntu 22.04 LTS.

2) Parameter setting: BERT-large is fine-tuned on ILDC using a batch size of 1 with a learning rate of 10^{-5} for 5 epochs, weight decay of 0.01 and warmup ratio of 0.06. After that, we extracted embeddings for both datasets as already told previously. Our proposed model was trained for 5000 iterations with a learning rate of 10^{-2}, and the discriminator was used with both the same learning rate and 10^{-4}. The used optimiser is SGD with a momentum of 10^{-4} and a dropout probability of 0.5 for fully connected layers. We performed a randomised grid search, by sampling ten sets of parameters over $\{0.25, 0.75, 1.5\}^3$ for what concerns $\{\beta_{\text{wordle}}, \beta_{\text{token}}, \beta_{\text{window}}\}$, while we fixed the other parameters as we mentioned before. For what concerns the wordle module we kept a window of two tokens to make the task simpler and the number of attempts fixed to 6. We also performed an ablation study, where we kept the same parameters as before and fixed $\{\beta_{\text{wordle}}, \beta_{\text{token}}, \beta_{\text{window}}\}$ to $\{0.75, 0.75, 0.75\}$.

B. Evaluation Metrics

Our evaluations are made through accuracy (both globally and by classes) and macro F1 score, involving ILDC as the training set and R3ad as the test set. For the metrics, we distinguished between source and target to evaluate the performance separately.

C. Results

1) Grid search: As we can note from table I, we obtain different parameters when we optimise for different metrics. In this context, this could imply that the model is robust to a change in parameters. We noted that different configurations performed similarly, and they are very close in terms of performance.

TABLE I
Results of Gridsearch

	Configuration $\{\beta_{\text{wordle}}, \beta_{\text{token}}, \beta_{\text{window}}\}$	Value (%)
Accuracy source	{0.25, 1.5, 0.75}	97.48
Accuracy source by classes	{0.25, 1.5, 0.75}	90.62
Accuracy target	{0.25, 0.75, 0.25}	85.11
Accuracy target by classes	{0.25, 0.75, 1.5}	38.99
F1 source	{0.25, 1.5, 0.75}	83.4
F1 target	{0.25, 0.75, 0.75}	35.26

TABLE II
RESULTS OF ABLATION STUDY

	Configuration	Value (%)	Improvement over naive architecture
Accuracy source	{token domain classifier, window domain classifier}	97.46	+0.03
Accuracy source by classes	{window domain classifier, wordle module}	90.64	+0.03
Accuracy target	{token domain classifier}	85.05	+0.26
Accuracy target by classes	{token domain classifier, window domain classifier, wordle module}	39.56	+0.63
F1 source	{token domain classifier, window domain classifier}	83.45	+0.54
F1 target	{window domain classifier, wordle module}	35.52	+0.20

2) Ablation study: Again, we experiment by searching for the best architecture for different metrics. As we can see from table II, all architectures perform similarly in the case of accuracies on the source domain. However, we obtain noticeable improvements over the other metrics. In particular, we obtain a gain of $+0.63$ over *accuracy target by classes*. This evidences that our additions provide benefits in terms of generalisation.

3) t-SNE: In our experiments, we also visualised t-SNE plots before and after training with all modules activated to understand the distribution of the embeddings. Before starting training, we can see that embeddings in the legal domain distribute quite well due to the fact that BERT-large was finetuned on the legal domain, as we can see from fig. 3. The embeddings in the defence domain, instead, mainly concentrate in one cluster except for a few points. We conjecture that this is due to some classes that overlap between entities of the two domains, such as organisation and people.

After training, as we can see from fig. 4, embeddings in the source domain clusterise even better. We also have an improvement for what concerns embeddings of the defence domain: however, we can hardly distinguish clusters. We believe that the finetuning on the legal domain caused the network to over-specialise on the L-NER task, making it hard to perform domain adaptation.

V. Conclusions

From the experiments we can make some deductions: fine-tuning on L-NER produces BERT embeddings over-specialised to recognise entities in the legal domain, as we can see from the clusters in the t-SNE visualisations, as a consequence there is no danger of catastrophic forgetting with the configuration we proposed. The model is still able to learn on the target domain, however, we can note a disproportionate degree of difference in the performance in the two domains. In order to overcome this problem, one could fine-tune higher layers of BERT, enhancing the contributions of the *wordle module* and of the *domain classifiers*. For what concerns the *wordle module*, we noticed that it is not able to learn well the task. This can be reconducted on the simplicity of the whole system and, similarly to what we stated before, it could reasonably be improved by unfreezing some of the BERT layers. For what concerns the *domain classifier*, we noticed that changing its learning rate is not effective. In conclusion, in this work we experimented with L-NER and transfer learning on a different dataset with little success: however, we gained insights on how fine-tuning BERT can lead to models which are over-specialised on a single domain.

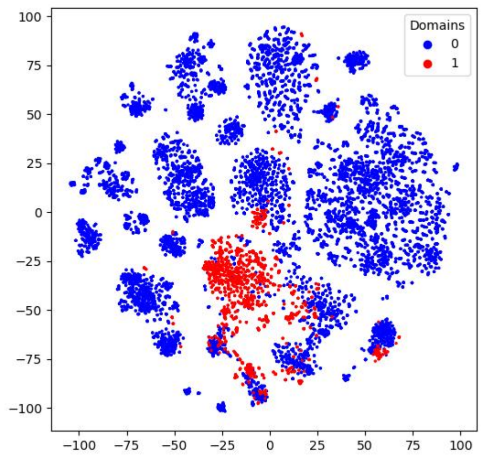

Fig. 3. t-SNE plot before training, domain 0 is the legal domain, while domain 1 is the defence domain

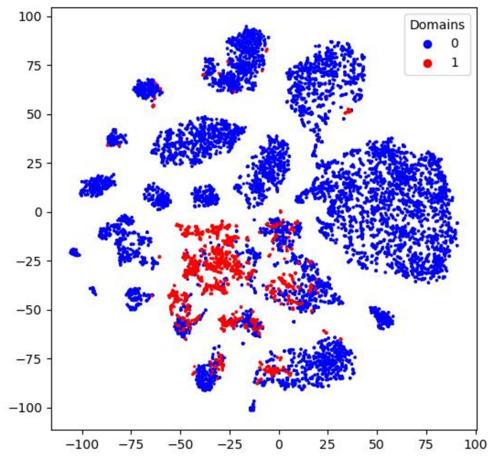

Fig. 4. t-SNE plot after training, domain 0 is the legal domain, while domain 1 is the defence domain

References

[1] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. March, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of machine learning research*, vol. 17, no. 59, pp. 1–35, 2016. 1, 2

[2] M. Crawshaw, "Multi-task learning with deep neural networks: A survey," *arXiv preprint arXiv:2009.09796*, 2020. 1, 2

[3] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos, "Legal-bert: The muppets straight out of law school," *arXiv preprint arXiv:2010.02559*, 2020. 1

[4] P. Kalamkar, A. Agarwal, A. Tiwari, S. Gupta, S. Karn, and V. Raghavan, "Named entity recognition in indian court judgments," *arXiv preprint arXiv:2211.03442*, 2022. 2

[5] I. Benedetto, A. Koudounas, L. Vaiani, E. Pastor, E. Baralis, L. Cagliero, and F. Tarasconi, "PoliToHFI at SemEval-2023 task 6: Leveraging entity-aware and hierarchical transformers for legal entity recognition and court judgment prediction," in *Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023)* (A. K. Ojha, A. S. Doğruöz, G. Da San Martino, H. Tayyar Madabushi, R. Kumar, and E. Sartori, eds.), (Toronto, Canada), pp. 1401–1411, Association for Computational Linguistics, July 2023. 2

[6] V. Malik, R. Sanjay, S. K. Nigam, K. Ghosh, S. K. Guha, A. Bhattacharya, and A. Modi, "Ildc for cjpe: Indian legal documents corpus for court judgment prediction and explanation," *arXiv preprint arXiv:2105.13562*, 2021. 2

[7] Aleph Insights and Committed Software, "R3ad - relationship and entity extraction evaluation dataset for defence domain." https://github.com/dstl/re3d, 2017. 3