# Multi-BERT: Leveraging Adapters for Low-Resource Multi-Domain Adaptation

**Parham Abed Azad**
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
parhamabedazad@sharif.edu

**Hamid Beigy**
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
beigy@sharif.edu

## Abstract

Multi-domain text analysis presents significant challenges, particularly in Persian name entity recognition (NER). Using a single model for multiple domains often fails to capture the specific features of different domains. That is why many scientists have focused on prompting chatbots for this issue. However, studies show that these models do not achieve remarkable results in NER tasks without proper fine-tuning while training and storing a chatbot is extremely costly. This paper presents a new approach using one core model with various sets of domain-specific parameters. By using techniques like LoRAs and pre-fix tuning, along with extra layers, we train each set of trainable parameters for a specific domain. This allows the model to perform as well as individual models for each domain. Tests on various formal and informal datasets show that by using these added parameters, the proposed model performs much better than existing practical models. The model needs only one instance for storage but achieves excellent results across all domains. This paper also examines each adaptation strategy, outlining its strengths, weaknesses, and the best settings and hyperparameters for Persian NER. Lastly, this study introduces a new document-based domain detection system for situations where text domains are unknown. This novel pipeline enhances the adaptability and practicality of the proposed approach for real-world applications.

## 1 Introduction

Named entity recognition (NER) is an essential part of natural language processing (NLP) that helps in many tasks like information extraction or question answering. Recently, NER has become even more important, thanks to the increased interest in NLP, which gave birth to many new challenges. One of the most pressing challenges is the adaptation of NER models to the ever-expanding text domains. This task becomes particularly difficult as model sizes and inference times increase. The dynamic nature of natural languages coupled with the diverse topics and contexts, presents a formidable challenge. Especially, for languages like Persian since there is a huge difference between texts found on formally written sites like Wikipedia and informally written texts from social media posts, even when those posts are made from official pages like universities. When we look closer, it becomes evident that models trained within a specific textual domain often achieve worse results when confronted with data from other domains. This performance gap shows that many sentences require different entity labels based on their specific context. For instance, consider the sentence "Tesla was robbed": in a scientific or historical context, "Tesla" would likely be tagged as a person, whereas in discussions related to business or economics, or within the context of a casual tweet, "Tesla" would be categorized as an organization. This ambiguity poses one of the primary challenges in accurately identifying entities, particularly within specialized fields such as medicine (Kundeti et al., 2016).

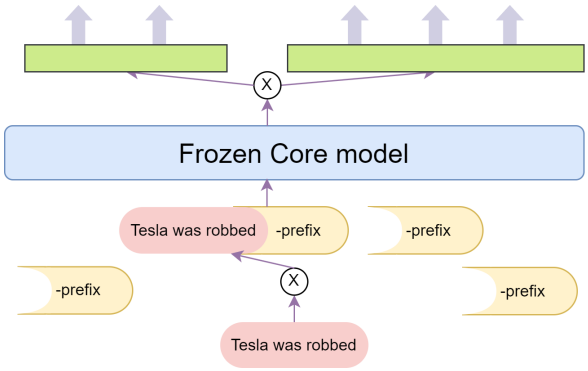



Figure 1: Each set of parameters belongs to exactly one domain but layers are often shared by a couple of domains.

Traditionally, the preferred approach was to train a single transformer model for all domains. The model would be trained in a way that performs



*Proceedings of the Tenth Workshop on Noisy and User-generated Text*, pages 108–116

well in all domains. However, while we praise these models for performing well without knowing the text domain, these models tend to perform worse than models trained on a single domain. This problem has stimulated the exploration of different strategies to overcome the barriers posed by adapting to multiple domains in NER. An additional challenge associated with this approach arises from the limitation of a single model to produce outputs in only one format. Certain domains may require different sets of labels, leading to the necessity for varied output formats tailored to each domain's needs. The adoption of multiple models to accommodate various domains introduces significant drawbacks, including resource-intensive requirements such as extensive RAM and storage constraints, as well as the time-consuming process of training each instance. Moreover, training a model for one task should inherently contribute to the understanding and performance improvement of another.

To deal with these challenges many have turned to prompt engineering of generative models such as OpenAI's ChatGPT. However, studies show that without fine-tuning, these models underperform fine-tuned models by a large margin in many NLP tasks such as NER (Abaskohi et al., 2024). Furthermore, training these models requires a lot of computational resources. and a huge sum of data to train the model. This problem is extremely exacerbated when we look at the languages that are suffering from a lack of well-labeled and clean data such as the Persian language. We delve deeper into the details of these models in section 2.

Therefore, a novel approach is proposed. This model offers an innovative solution to address these challenges by leveraging adapters. We incorporate specific parameters for each domain and create individual output layers to produce distinct outputs for each domain. Subsequently, the added parameters and the output layers are trained for each domain. Therefore, each set of these parameters and layers is only used for certain domains. This allows the model to perform perfectly in all domains. Given access to a robust pre-trained model, the core model is frozen; however, it can also be trained during the training process if a pre-trained model is unavailable, the layers are shared between multiple domains, and the adapters are each only used for a single domain. Remarkably, even when facing limited data availability, we observe a significant performance boost compared to other models. Finally, we introduce an innovative approach for situations where the domain is unknown.

The rest of this paper is organized as follows: section 2 gives a brief overview of the related projects that try to solve these issues. Thereafter, section 3 explains the proposed architecture of the multi-Bert model and the model will be thoroughly evaluated in section 4. Moreover, section 5 will propose a novel pipeline that deals with the issues that arise from not knowing the exact domain of the texts, while section 6 concludes the paper and section 7 talks about the future possibilities.

## 2 Related work

Named entity recognition has always been a popular task in NLP. Many new papers like PUnified-NER advocate for customizing and training a generative LLM model capable of understanding diverse text domains and label sets (Lu et al., 2023). By incorporating a lot of information into prompts and leveraging extensive training data, this model demonstrates a remarkable capability to label various data types with diverse labels. However, other papers focus on the development of template-free models utilizing few-shot learning. These studies introduced models that, with a minimal set of labeled examples (typically 16, 32, or 64), can adeptly label texts (Wang et al., 2022; Lu et al., 2023; He et al., 2023; Ma et al., 2022). However, these papers still fine-tuned the model. In fact, it is shown that for some NLP tasks like NER fine-tuning the model is a crucial step and solely relying on prompt engineering results in subpar results (Li et al., 2023). Furthermore, Our experiments with ChatGPT on Arman and ParsTwiNER datasets have resulted in much worse performance compared to Bert models. This is in line with other scientific research done with LLMs like Abaskohi's benchmarking of ChatGPT which achieved decent results on tasks like question answering while getting extremely low results on token classification tasks. (Abaskohi et al., 2024).

At the forefront of Persian NER, current state-of-the-art models include BeheshtiNER (Taher et al., 2020) and ParsBERT (Farahani et al., 2021). a BERT model fine-tuned for NER tasks. Recent advancements in Persian NER have predominantly focused on fine-tuning models to cater to diverse domains. Outstanding examples of this strategy include ParsTwiNER (Aghajani et al., 2021), a BERT model fine-tuned for informal and formal texts.

However, as seen in this paper, while the model outperforms the original ParsBert on informal text, it underperforms on the formal dataset, Arman. Another example is Hengam, a BERT model tailored for token classification in the tagging of formal and informal texts. As seen, these models exhibit notable drawbacks, such as prolonged training times and diminished performance in previous domains when adapting to new ones. The need for more efficient and adaptable models remains an ongoing concern within the landscape of Persian NER research.

## 3 The proposed method

To mitigate the challenges posed by time and size constraints inherent in employing multiple models, a model called multi-Bert is presented. In multi-Bert, a single pre-trained model is completely frozen, while multiple sets of additional parameters and layers are integrated into the model. This configuration allows for the generation of diverse results from a single model. Moreover, this design facilitates training for specific tasks without modifying the underlying base model, thereby safeguarding the performance of one task from affecting another. However, while the domain-specific parameters are completely separate from each other, we use pre-training for each set of parameters on other sets of data. This approach allows us to leverage any correlating information that can aid the model's performance. As seen in figure 1, the model allows the selection of task-specific parameters during inference, tailoring the model's behavior to the requirements of each individual task.

A significant advantage of multi-Bert is its efficient use of adapters compared to traditional fine-tuning methods. Adapters accelerate the training process (He et al., 2021), reducing the time required for each epoch and enabling model convergence in fewer than 10 epochs. This efficiency allows us to employ a two-step training approach: first, pre-training one set of parameters on all available data, and then fine-tuning a copy of these parameters for each specific task of interest. By leveraging task-specific data, we can effectively utilize cross-task information during fine-tuning.

To incorporate these parameters we utilize adapters. After exploring various methods and adapters the most effective techniques were selected. The first chosen adapter is Prefix-tuning (Li and Liang, 2021). Prefix tuning is a technique where a small set of learnable parameters, known as a prefix, is embedded directly into the input of all layers of a pre-trained language model. This allows the model to rapidly adapt to task-specific information without the need to fine-tune all the model parameters. The prefix acts as a continuous task-specific vector that can influence the behavior of the model across all layers, providing a lightweight and efficient way to customize large language models for specific tasks. It has been shown to achieve comparable performance to full model fine-tuning while requiring the tuning of only a tiny fraction of the parameters By leveraging the structured nature of prompts, this approach facilitates prompt-driven learning, a crucial aspect in multi-domain scenarios. On the other hand, for adapters, we employ the well-known Low-rank adaptation method, also known as LoRA (Hu et al., 2022). This method yields comparable results by incorporating learnable parameters into the model layers. LoRA focuses on preserving adaptability without compromising the integrity of the base model. By adding parameters to each layer without introducing new ones, LoRAs have emerged as highly reliable adapters. Their efficiency lies in seamlessly integrating new parameters into existing layers, yielding impressive results within a short time frame, and facilitating straightforward merging of the new parameters with the existing layers.

Additionally, a classification layer is introduced based on the required number of classes. In cases where tasks share the same output structure, both the size of output and the specific labels, this layer can be shared among them. Conversely, for tasks with differing output structures, we accommodate multiple final layers tailored to each task's unique requirements. This streamlined approach not only addresses the challenges associated with multiple models but also provides flexibility in adapting to diverse task requirements. The effectiveness of multi-Bert is validated through comprehensive evaluations utilizing various parameter addition methods and task-specific classification layers.

We use a fine-tuned core model, ParsBert which is arguably the best pre-trained Bert model. There are a lot of different models pre-trained for the Persian language and each one can be used. However, based on our calculations ParsBert performs the best for the NER tasks. Therefore, ParsBert was used in this study and due to the high performance of this model, it was frozen throughout the training

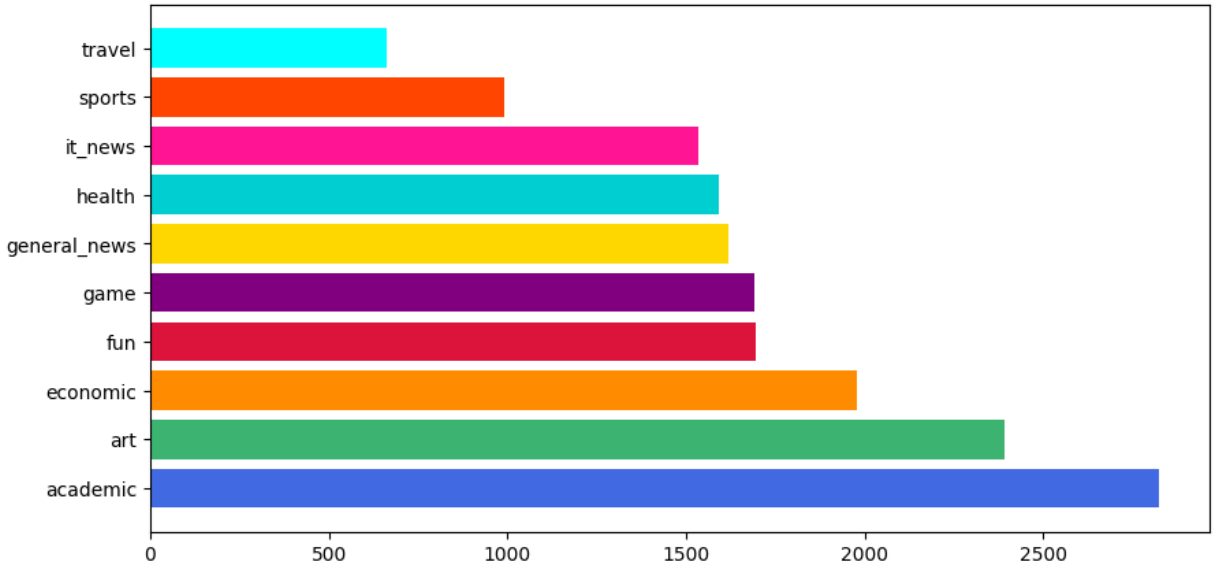



Figure 2: The size of the data in each domain of text greatly differs from one another, which results in massive challenges.

steps of the multi-Bert model.

There are a few steps to train the model, firstly, for each domain, exactly one adapter is introduced, and for each set of adapters, that have the same output template, a classifier header is included. Initially, one adapter is trained on all available data associated with that classifier excluding the domain of interest for a couple of epochs to ensure the adapters have all the correlating knowledge from other domains and the classifier is properly tuned. Subsequently, this adapter is replicated across all other adapters of the same classifier and fine-tuned on each domain separately, until convergence and before over fitting.

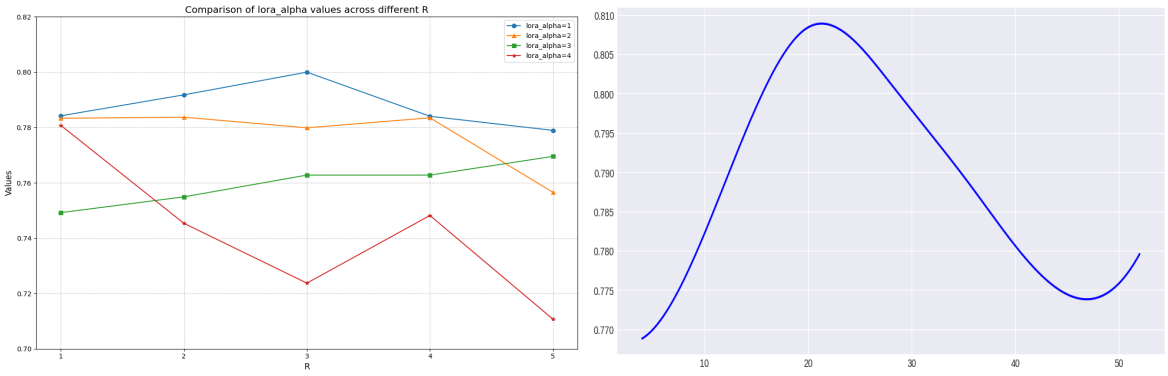
The classifier is only trained when the adapter is being pre-trained since this step includes all of the data associated with that particular output scheme. Furthermore, this layer is frozen during the fine-tuning of the adapters this helps make the overall training to be shorter and helps preserve the knowledge of the adapters from inference. After the training process is complete, there is only one core model with multiple headers and many domain-specific parameters. Therefore, we have multiple models each tailored for a single domain that can perform separately while they share much of their architecture. The subsequent section outlines the implementation and workflow of the proposed model, showing its efficiency and adaptability in handling multi-domain NER tasks.

## 4 Evaluation and Results

The proposed model is evaluated by using three different distinct datasets, each chosen to represent diverse unique challenges in the realm of NER and to test the performance of our model in dealing when faced with different challenges. We also introduce important baselines to show the effectiveness of our model. Finally, we compare the results of all the models and discuss the hyperparameters and advantages of prefix-tuning and LoRAs.

### 4.1 Datasets

To evaluate the aforementioned strategy the models are tested on three distinct datasets. For the first two datasets, we focus on the classic formal versus informal NER tasks, we utilize the Arman dataset for formal entities, and ParstwiNER (Aghajani et al., 2021), for the informal ones. This dataset serves as a benchmark for the adaptability of our model across standard formal and informal contexts characterized by minimal noise. This is very important since in many datasets all the text does fall into these domains. For instance, if we have a close look at the data on Twitter's more established accounts we see that many tweets are written perfectly and cleanly whether in a formal language or an informal one. However, these two datasets are extremely standard, they are both based on the CONLL format, have 21 entity types, and lack considerable errors or use of niche grammar which makes them very similar to the majority of the text the core model is trained on. Hence, the results on




(a) The best score is from the blue line

(b) F1 peaks at 22 and 23 and falls after that.

Figure 3: All values are F1 scores of training a pre-trained model with the adapter for 2 epochs.

these two datasets differ from when datasets with noisy data taken from sites like Twitter are used.

That's where the third dataset, ParsNER (Asgari, 2021), comes in, ParsNER, This dataset consists of a huge amount of noise, whether it is words that are tagged inconsistently or general script errors. More importantly, the labels of this model are different from the previous datasets with only nine tags and a "MISC" tag that is supposed to represent any other tag, and probably any other dataset, since it's not based on a standard. The data is taken from posts on Twitter pages reflecting different topics. Thus, the data is clustered and grouped in different domains. These domains are extremely different from each other and as we mentioned at the start of the paper, the tagging will be greatly influenced by the topic at hand as a word like "Iran" is probably a "loc" when we are talking about travel and an "org" when we are talking about economics. This feature turns the differences in domain huge as we will see in later results that models that are specialized in the domain greatly outperform general models.

Moreover, the number of entries in each domain differ from one another you can see the number of entries in each domain in the figure 2. Normally, we would not be able to share one model for this dataset with the previous ones due to these huge differences. Still, with multi-Bert, we will use our model to achieve state-of-the-art results across all of our domains and datasets to show that this model can truly be a solution fit for all problems. Therefore, we have two sets of data with different outputs each one has other domains that we need to focus on and we will show that one instance of our model can give state-of-the-art results across all of these domains and datasets.

### 4.2 Baselines

For the baseline, we introduce two models that we expect our model to perform between them. Firstly, our lower bound baseline is a general model that is trained on all of the domains however since a model cannot give outputs in two different formats like our multi-Bert we train two general models one for the first two datasets and another for all the domains in the ParsNER dataset. These general models are trained by fine-tuning our pre-trained core model on the concatenation of all domains. However, we also design an upper-bound baseline. We fine-tune the core model on all of our data and fine-tune it on a single domain, we do this twelve times for each single domain. This is an extremely time-consuming experiment and the result is twelve huge models that are not a feasible solution. However, this does give us the best possible solution. Furthermore, ChatGPT is used to tag the sentences in the formal and informal datasets. However, a simple prompt is used that gives the model the desired tags and asks the model to tag each word. Using approaches like few-shot learning or training the model might achieve greater results but those require a huge sum of data and computational power, that the abstinence of it, is the main problem we are trying to solve.

### 4.3 Hyper-parameters setting

One of the main challenges of using adapters in general is the complex parameters. In this paper, we used an approach that closely resembles grid-search. Firstly we set all parameters to every number that is apart from each other eight(4, 12, 20, e.g) after finding the best performing sets of parameters, we adapt the model to all possible parameters in the range. After many experiments on our different

| Model | Domains | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | acad | art | econ | fun | game | news | med | it | sport | travel |
| Gen-Bert-9 | 82% | 66% | 70% | 69% | 78% | 79% | 83% | 86% | 81% | 76% |
| Spec-Bert | 86% | **87%** | **93%** | 90% | **96%** | 93% | **95%** | 93% | **95%** | 90% |
| Multi-lr | 70% | 78% | 80% | 86% | 90% | 87% | 86% | 83% | 92% | 87% |
| Multi-pre | **90%** | **87%** | 86% | **92%** | 91% | **97%** | 93% | **94%** | 93% | **95%** |

Table 1: The F1 Scores of the general Bert is overshadowed in any domain but the difference is much more visible in smaller domains

datasets, we came to the final conclusion that the best number of tokens to add to our prompt is 22. Adding fewer tokens to each input layer results in worse performance while adding more tokens leads to no positive change in the model performance, if not worse, and only leads to much longer training time. For the LoRA model, we also did the same thing but the only difference is that there are two parameters and the result is dependent on their combinations. Thus, Moreover, we use a batch size of 16 with a learning rate of 0.0001 as it has proven to give the best results.

As we see in the figure 3 the performance of the model rises up to the 18 parameters and starts falling significantly which makes choosing the 22 an easy choice. However, it's a little more complicated for the LoRA model since the results of the model for each value of R or alpha are dependent on the value of the other one. We came to the conclusion that for our named entity recognition task the best combination is LoRA alpha and r of 1 and 3 respectively. With these hyperparameters, the prefix model adds 468,490 parameters, and the LoRA model adds 136,714 parameters to the model, which constitute 0.38% and 0.11% of the model parameters, respectively. Since the base model has 124 million parameters, the added parameters are less than 1% of the model parameters, and having a few of these adapters is very low-cost.

| Model | parameters | Save/trainable |
|---|---|---|
| Bert-21 | 117,722K | 100% |
| Multi-Bert-lr | 118,070K | 0.294% |
| Multi-Bert-pre | 119,522K | 1.503% |
| ChatGPT | 175B | 100% |

Table 2: Only a small percent of the parameters are trained and saved for Multi-Bert

### 4.4 Training details

In this section we will discuss the details of the models and the training procedure. The table 2 shows the number of parameters. The models were trained on a t4x2 for 30 epochs and saved the best model.

### 4.5 Results

As we see in the final results at table 3 and table 1 the general model under-performs in every domain. It is clear that the general model performs better at Formal v. Informal task compared to the 10-domain task for multiple reasons. Firstly, the data between the two classes are more even, we see that in the second general model, the domains with much smaller datasets are clearly forgotten for the sake of the bigger domains. Secondly, the more the number of our domains is, the harder it becomes for the model to adapt to all of them. We see that in the results of the domains that have much less data compared to their competitors, the model gets over-fixated on the other domains and greatly under-performs in these domains while doing relatively well in the domains with more data. When we look at the F1 score of the specialized models we see that even though each one of them is trained on all of the data and specialized on a single domain they do not outperform our model by a huge margin, in fact, the multi-Bert with prompt-tuning outperforms fine-tuning a model on multiple instances by a small margin. Another important observation is that prompt-tuning outperforms LoRA and is championed as the best way to create this model. This is due to the problem of limited data. Adapters require more data to train effectively. For this simple reason, we see that LoRA gives us results close to the fine-tuned ones, but is greatly overshadowed by prompt-tuning for the domains with much smaller datasets.

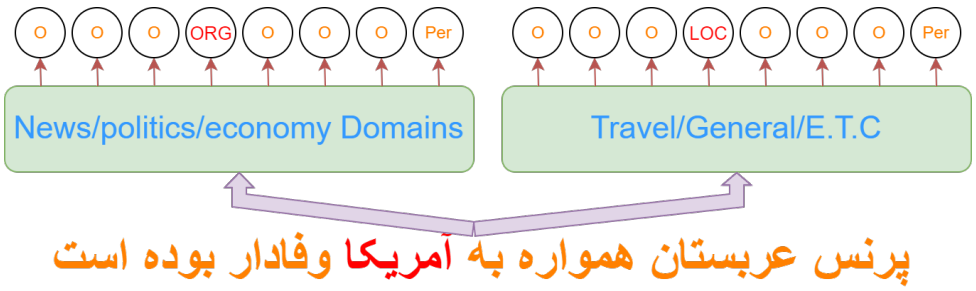



Figure 4: The models specialized on News, politics and economy get the tag of US right. Text from the political domain of ParsNER. Text translation: The Saudi prince proved to be a loyal ally to the United States.

| Model | Arman | ParstwiNER |
|---|---|---|
| General-Bert-21 | 95.2% | 75.4% |
| Spec Bert | **99.6%** | 81.7% |
| Multi-Bert-lr | 99.4% | 80.8% |
| Multi-Bert-pre | 99.3% | **83.1%** |
| ChatGPT | 67.8% | 55.7% |

Table 3: Multi-Bert achieves similar results to a whole specialized Bert.

### 4.6 Discussion

So the model is getting better results, but why and where are these improvements? To answer these essential questions in this section the domain models are tested on a particular example from the ParsNER dataset in the general news section. The translation of the sentence goes "The Saudi prince proved to be a loyal ally to the United States". The word United States has a huge ambiguity here, if the loyalty is to the land of the country it needs to be labeled as "LOC", however, if the point is to be loyal to the government of the country the label would be "ORG". To us, humans, labeling this sentence might not be that hard, after all from the tone and the context we might be able to understand that the context of the sentence is politics. But this sentence proves to be exceptionally hard for the model, as seen in the figure 4 not only do the travel domains get this answer wrong but general models and others such as the ones designed for IT news also get this sentence wrong while the models that know the context of politics, economics or even general news get it right. This also outlines that a small boost goes a long way as seen in some specialized models.

## 5 Document-based classifier pipeline

In this section, we address the challenges posed by the absence of domain knowledge and propose an innovative solution to overcome this obstacle. Fortunately, determining the domain of a given text becomes relatively straightforward when the text is sufficiently long or when multiple samples are available. Nevertheless, feeding multiple samples simultaneously to a model is impractical, as it may lead to unwanted interference among distinct entries. To tackle this issue, we introduce a new pipeline by fine-tuning a new set of parameters to our core model.

To achieve this objective, we aggregate every set of elements (for example 8 elements) and assign them a label representing the domain of the data. Subsequently, we shuffle the data from all domains and train a new adapter for the model with the additional parameters tailored for a text classification task. Upon completion of the training process, we construct our pipeline as seen in the figure 5. When employing the model for inference—whether it involves tagging a series of comments on a website, tweets within a Twitter thread, or processing a lengthy book—we provide 512 tokens from the text to the model to find the domain and based on the identified domain, we apply the respective parameters from the core model to obtain the final results.

It is important to note that since we utilize the core model already employed in our token classification tasks and entirely freeze the core model during the classifier training, this pipeline does not adversely impact the main token classification models. To train, we concatenate each 8 input rows as one input. However, we only concatenate up to a length of 512. Therefore, if the sum size of 5 elements exceeds 512 we only concatenate 5 elements

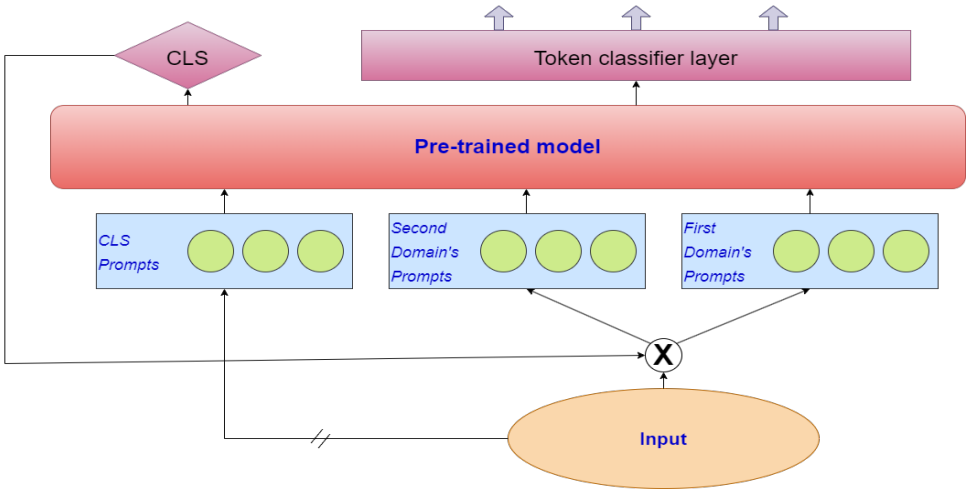



Figure 5: One forward pass determines the domain of a set which then can be used for each single input.

and cut the first 512 tokens of it. Consequently, each group of inputs turns into one input with the label of the dataset they are picked from. Then, we mix and shuffle all of the data and train and evaluate the model on all the data.

For the formal and the informal datasets, we get an accuracy of 100%. This is reasonable considering the distinctions between these two datasets. However, when we look at the ParsNER dataset we get an accuracy of 97% which is decent since we have 8 different classes. Moreover, even if this model fails to predict the right domain of the text, it does not guarantee a wrong final output as the chosen version may generate correct results. In fact, the decided domain is probably extremely close for this mix-up to happen. Hence we can use this pipeline to use the collection of texts to decide on their context and then process them normally one by one.

## 6 Conclusions

This paper proposed the multi-Bert model. This model is designed to perform well for all domains with any set of outputs. This is thanks to the deliberate design of this model by adding parameters for each domain and output layers for different sets of outputs coupled with the faster training time. This design allowed the model to perform the task on all domains perfectly. Moreover, this paper evaluated the proposed model on Arman (a formal dataset), ParstwiNER (an informal dataset), and the ParsNER, a collection of ten datasets from different contexts with large amounts of noise. The results proved that this model performs as well as the state-of-the-are for each domain, if not better. In addition, we also proposed a pipeline that can decide the domain of the data when a small set of sentences are available. We observed that if we use the model for sets of 8, it could understand the formality of the inputs completely with a 100% accuracy and can classify the exact domain of the news with an astonishing accuracy of 97% for the ParsNER dataset.

## 7 Future works

There is much to do in the future as this paper is only one step toward dealing with multi-domain problems. Firstly, creating a Multi-Bert with the newer proposed ModernBERT might achieve greater results (Warner et al., 2024). Secondly, the effectiveness of chatbots should be tested in multi-domain settings with fine-tuning. Due to limited computational resources, we only tested prompt engineering for generative LLMs but as seen in other papers generative LLMs do not have satisfying performance without proper fine-tuning (Abaskohi et al., 2024). While no one has actually fine-tuned these models for NER, fine-tuning these models will probably give us better results compared to smaller models like BERT. However, this task needs a huge amount of computational power. Last but not least, the proposed method approach should be tested for other low-resource NLP tasks such as question answering. Domain knowledge becomes even more important in generative tasks such as translation and question answering since the generated text also needs to be in the domain of the incoming text. However, in such research, using BART or generative LLMs such as LLAMA might give better results.

## Limitations

This work is limited by the smaller architectures of the Bert model. While the smaller size and parameter count helps us fine-tune the model with low computation power it limits the results we are able to get compared to huge generative models. Moreover, the proposed approach is only tested for the Persian language and only for the NER task. Furthermore, the tests on ChatGPT are done with a simple prompt, while our results are in line with some of the found research in this field engineering greater prompts or using few-shot approaches might result in a higher F1 score. However, due to the huge difference in performance compared to the fine-tuned models used in this paper it is extremely likely that any prompt would achieve a accuracy close to the trained Bert models.

## References


Amirhossein Abaskohi, Sara Baruni, Mostafa Masoudi, Nesa Abbasi, Mohammad Hadi Babalou, Ali Edalat, Sepehr Kamahi, Samin Mahdizadeh Sani, Nikoo Naghavian, Danial Namazifard, et al. 2024. Benchmarking large language models for persian: A preliminary study focusing on chatgpt. *arXiv preprint arXiv:2404.02403*.

MohammadMahdi Aghajani, AliAkbar Badri, and Hamid Beigy. 2021. ParsTwiNER: A corpus for named entity recognition at informal persian. In *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, pages 131–136.

Majid Asgari. 2021. Parsner. https://github.com/majidasgari/ParsNER.

Mehrdad Farahani, Mohammad Gharachorloo, Marzieh Farahani, and Mohammad Manthouri. 2021. ParsBert: Transformer-based model for Persian language understanding. *Neural Processing Letters*, 53:3831–3847.

Kai He, Rui Mao, Yucheng Huang, Tieliang Gong, Chen Li, and Erik Cambria. 2023. Template-free prompting for few-shot named entity recognition via semantic-enhanced contrastive learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–13.

Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jia-Wei Low, Lidong Bing, and Luo Si. 2021. On the effectiveness of adapter-based tuning for pretrained language model adaptation. *arXiv preprint arXiv:2106.03164*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank adaptation of large language models. In *International Conference on Learning Representations*.

Srinivasa Rao Kundeti, J Vijayananda, Srikanth Mujjiga, and M Kalyan. 2016. Clinical named entity recognition: Challenges and opportunities. In *Proceedings of IEEE International Conference on Big Data*, pages 1937–1945.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Zongxi Li, Xianming Li, Yuzhang Liu, Haoran Xie, Jing Li, Fu-lee Wang, Qing Li, and Xiaoqin Zhong. 2023. Label supervised llama finetuning. *arXiv preprint arXiv:2310.01208*.

Jinghui Lu, Rui Zhao, Brian Mac Namee, and Fei Tan. 2023. PUnifiedNER: A prompting-based unified ner system for diverse datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13327–13335.

Ruotian Ma, Xin Zhou, Tao Gui, Yiding Tan, Linyang Li, Qi Zhang, and Xuanjing Huang. 2022. Template-free prompt tuning for few-shot NER. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5721–5732.

Ehsan Taher, Seyed Abbas Hoseini, and Mehrnoush Shamsfard. 2020. Beheshti-NER: Persian named entity recognition using BERT. *arXiv preprint arXiv:2003.08875*.

Liwen Wang, Rumei Li, Yang Yan, Yuanmeng Yan, Sirui Wang, Wei Wu, and Weiran Xu. 2022. InstructionNER: A multi-task instruction-based generative framework for few-shot ner. *arXiv preprint arXiv:2203.03903*.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, et al. 2024. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. *arXiv preprint arXiv:2412.13663*.