AaltoNLP at SemEval-2022 Task 11: Ensembling Task-adaptive Pretrained Transformers for Multilingual Complex NER

Aapo Pietiläinen    Shaoxiong Ji
Aalto University
{aapo.pietilainen; shaoxiong.ji}@aalto.fi

Abstract

This paper presents the system description of team AaltoNLP for SemEval-2022 shared task 11: MultiCoNER. Transformer-based models have produced high scores on standard Named Entity Recognition (NER) tasks. However, accuracy on complex named entities is still low. Complex and ambiguous named entities have been identified as a major error source in NER tasks. The shared task is about multilingual complex named entity recognition. In this paper, we describe an ensemble approach, which increases accuracy across all tested languages. The system ensembles output from multiple same architecture task-adaptive pre-trained transformers trained with different random seeds. We notice a large discrepancy between performance on development and test data. Model selection based on limited development data may not yield optimal results on large test data sets.

1 Introduction

SemEval-2022 shared task 11: MultiCoNER (Malmasi et al., 2022b) was about complex Named Entity Recognition (NER) in the multilingual context. Transformer-based models such as BERT (Devlin et al., 2019) have reached high accuracy in standard NER tasks. However, the models still struggle with complex and code-mixed named entities (Meng et al., 2021; Fetahu et al., 2021). The shared task tries to address the challenges regarding complex NER in a multilingual context. Out of the 13 available tracks, we focus on 5 monolingual tracks. We build models for Bangla, English, Farsi, German, and Korean.

Our strategy is to build an ensemble approach to increase accuracy compared to the baseline model. To tackle multilingualism, we build an approach that starts from the same XLM-RoBERTa (Conneau et al., 2019) encoder and after fine-tuning, achieves good performance across languages. We propose two ensembling approaches: (1) naive ensembling and (2) end-to-end (E2E) ensembling. Naive ensembling is a test-time ensembling where class scores from individually trained models are added together. E2E ensembling trains two encoder models jointly by concatenating their results and passing the concatenated predictions to the final layer. In addition to these strategies, we adapt the encoder model to data context using continued pre-training (Gururangan et al., 2020).

We discover that naive ensembling produces good results with few models and by just using different random seeds for training. We also discover that performance can drastically vary between data sets and model selection based on development data may not yield good results on large test sets. Our final rankings are 11th for Farsi, 12th for Bangla, 13th for German, 15th for Korean, and 25th for English. For each language, our ranking is on the lower half of the participants. The code for our submission has been released 1.

2 Background

In NER, complex and ambiguous entities are hard to classify correctly. Out of the 13 available tracks, we participated in the monolingual tracks for Bangla, English, Farsi, German, and Korean. For training, we used only the competition data sets (Malmasi et al., 2022a) provided by the organizers. The task is to detect named entities in given sentences. Figure 1 contains an example of the task setting. This example can be considered complex, as Madagascar is ambiguous as it could refer to the country or the movie.

Transformer-based models (Vaswani et al., 2017) such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) produce state-of-the-art results on variety of natural language processing (NLP) tasks including NER. Transformer models trained

1Code is available here https://github.com/aapop/multiconer_AaltoNLP
Figure 1: An example of NER task. A sentence from the development data set and the corresponding labels.

![Table](image)

### System overview

The motivation for our system is to produce an approach that can be applied across languages. Initial testing with the baseline system provided by the organizers suggested that XLM-R_{large} (Conneau et al., 2019) produces comparable or even better performance than tested monolingual pretrained models for targeted languages. Therefore, we resort to building on top of a multilingual encoder model XLM-R_{LARGE} and the provided baseline system.

Inspired by results using task-adaptive pretraining (Gururangan et al., 2020), we start by training task-adaptive language models. We hypothesize that task-adaptive pretraining could provide two sources of improvement: (1) adapting to the data domain and (2) improving the capability in the specific language. Task-adaptive pretraining is conducted for the XLM-R_{LARGE} using the provided training data, which we prepare for pretraining by removing labels and constructing line-by-line sentences. The pretraining is applied to each language separately yielding us five encoder models named koala/xlm-roberta-large-XX. The XX is replaced with the corresponding language identifier. The pretraining is continued from the XLM-R_{large} checkpoint with only the preprocessed task data. The pretraining objective remains the same. The models are available at the Huggingface model repository. In this paper, we refer to the models as koala-XLM-R_{large} interchangeably for all languages.

After obtaining the enhanced encoder models, we propose two ensemble learning approaches, i.e., naive ensemble and E2E ensemble.

#### 3.1 Naive ensemble

As the first ensemble strategy, we combine predictions from multiple independently trained models at test time. Randomness and choosing a random seed can have a significant effect on the model training and final accuracy. Based on that, our hypothesis was to leverage this randomness to increase the accuracy.

To ensure applicability in languages, where a limited number of available models are available, we train the models using the same architecture and only vary the random seed associated with training. The koala-XLM-R_{large} is used as the encoder model for each language respectively. After the encoder model, we use the linear layer as the prediction head and apply softmax transformation to obtain class probabilities. The models were trained separately and optimized with AdamW optimizer (Loshchilov and Hutter, 2018) and negative log-likelihood loss function (NLLLoss).

The ensembling is conducted during test time. We simply add the class probabilities together from each of the models:

$$ y^* = \text{argmax}_j \sum_n p_{nj}, \quad (1) $$

where $p_{nj}$ is the probability of model $n$ for the class $j$. The predicted class $y^*$ is the class with the highest sum of probabilities. For the ensemble weights,
we also tried entropy minimized ensemble (Wang et al., 2021), which is based on the intuition that
good ensemble weights should decrease entropy
(Shannon, 1948; Wang et al., 2020). The algorithm
is elaborated in Algorithm 1. This approach did not
provide noticeable improvement in our tests and
thus, we proceeded with uniform weights.

Our ensemble model consists of four distinct
models. The number of four models was chosen as
a compromise on the trade-off between accuracy
and inference time. The model system is illustrated
in Figure 2.

3.2 E2E ensemble

Our second ensemble strategy centered around
the idea of jointly training the models. The idea is
to include sub-networks (encoder models) in the
model, ensemble their predictions, and optimize
the loss function jointly. Our system uses koala-
XLM-R_{LARGE} and XLM-R_{BASE} as the encoder models.

The input data is passed to both encoder models.
After that, we apply linear and softmax layers. For
the second encoder model, the linear layer outputs
2 * num classes scores. We made this deci-
sion to add variation between the two sub-networks.
The choice to use twice the amount of parameters
was arbitrary. We tried a different number of pa-
rameters but the performance seemed similar.

After outputting the scores from the two sub-
networks, the scores are concatenated together.
Also, summation was tested when using same-sized
outputs, but concatenation seemed to perform better.
After that, the concatenated scores are passed
to linear and CRF layers. The architecture of E2E
ensemble model is illustrated in Figure 3.

The E2E ensemble model consists of two sub-
networks. We tried using also three and four sub-
networks but adding more models did not improve
the performance. Therefore, we settled on using
two sub-networks, which also help with the com-
putational burden. We experimented with different
sub-network architectures and noticed that using
different encoder models provided the best perfor-
mance despite varying the linear layer size. Also,
using koala-XLM-R_{LARGE} instead of XLM-R_{BASE} as
the second sub-network did not improve accuracy.
Hence, the smaller model was used.

The model was optimized using AdamW optimizer
with Viterbi loss as the loss function.

4 Experimental setup

Our experimental setup was based on the provided
baseline model (Malmasi et al., 2022b) and the data
splits (Malmasi et al., 2022a). Models were trained
only using the training data and the best model was selected based on performance on development data. After selecting the best model, we did not do any training with development data. For continued pretraining, we preprocessed the training data. The data were converted from CoNLL format into line-by-line text. Hence, we removed the labels and reconstructed the sentences into a single line of text data.

We selected suitable hyperparameters based on performance on the development data. We selected two learning rates, first for the encoder model and a second one for all subsequent layers (decoder). We set the encoder learning rate to $10^{-6}$ and the decoder learning rate to $10^{-3}$. To prevent overfitting, we used a dropout rate of 0.1. We also used early stopping with development data. The training was stopped when accuracy on development data had not increased in the last three epochs. We selected batch size based on the hardware constraints. For baselines and sub-models in the naive ensemble, we used batch size 64. For E2E ensemble, batch size is 20.

The main Python libraries we used are PyTorch, PyTorch Lightning, AllenNLP, and Transformers. Further details available in the repository.

The performance in this shared task is evaluated using prediction accuracy on unseen test data set. The teams are ranked by their macro-averaged F1 classification score.

## 5 Results

The official metrics were based on the evaluation phase with test data. The participants were ranked by their Macro F1 scores. The performance of built models on test data is shown in Table 1. The best score is bolded and is the model used for ranking. It can be seen that the naive ensemble model performs better than the best individual model across all languages. E2E ensemble model shows mixed results and performance varies run by run. It outperforms naive ensemble on Korean and Farsi test data.

We noticed a rather large discrepancy in accuracies between the development and test data set. The details of model performance on development data are presented in Table 2. Compared to the results on test data, the performance on development data was significantly better. The discrepancies between performance on development and test data suggest that model selection on such limited development data yields sub-optimal results.

Our systems suffer with some of the classes. As can be seen from Figures 4 and 5, PROD and CW classes have significantly lower accuracy than other classes. Performance on GRP, LOC, and PER classes is much higher. Our systems suffer with the no-tag-class O, which has the majority of the misclassifications.

### 5.1 Task-adaptive pretraining

We hypothesized that task-adaptive pretraining would adapt the multilingual encoder model into the data and language domains. We tested the performance by training the model four times using different random seeds and comparing the average micro F1 score between XLM-R\textsubscript{large} and koala-XLM-R\textsubscript{large} models. The results of task-adaptive pretraining are elaborated in Table 3. The improvements on this task are small and compared to the original paper, in which, the accuracy increase was considerably larger on some data sets. For German, the accuracy even decreased significantly. The decrease is mainly caused by random variation as one of the four models for koala-XLM-R\textsubscript{large} achieved only accuracy of 0.81 when all other models (XLM-R\textsubscript{large} and koala-XLM-R\textsubscript{large}) were in the range of $0.85 - 0.88$.

<table>
<thead>
<tr>
<th>Model</th>
<th>en</th>
<th>ko</th>
<th>fa</th>
<th>de</th>
<th>bn</th>
</tr>
</thead>
<tbody>
<tr>
<td>XLM-R\textsubscript{large}</td>
<td>0.836</td>
<td>0.788</td>
<td>0.757</td>
<td>0.873</td>
<td>0.738</td>
</tr>
<tr>
<td>koala-XLM-R\textsubscript{large}</td>
<td>0.837</td>
<td>0.793</td>
<td>0.761</td>
<td>0.850</td>
<td>0.746</td>
</tr>
<tr>
<td>Improvement</td>
<td>0.1%</td>
<td>0.6%</td>
<td>0.5%</td>
<td>-2.6%</td>
<td>1.1%</td>
</tr>
</tbody>
</table>

Table 3: Average micro F1 scores for four models. Models trained with different random seeds.
As discussed earlier, we tried to improve our ensembling strategy using entropy minimization. The results are reported in Table 4. Despite testing different approaches, no sign of improvement is present. The testing was conducted on the Bangla development data set. Our setting differs from the original authors as we are not using off-the-shelf language adapters. We probably have too few models and they are not diverse enough.

6 Conclusion

From our efforts, we conclude that naive ensembling improves accuracy with just four models of the same architecture trained with different random seeds. The ensemble of four models outperforms the best individual model across all the tested languages. The E2E ensemble model can provide good accuracy, but the results vary drastically between runs. Task-adaptive pretraining, which has in some cases improved accuracy significantly, yielded only a slight improvement in this task. We also notice a large difference between performance in development and test data. With such limited development data and a large test set, model selection on solely development data yields sub-optimal results. More attention should be paid to model selection and model’s generalizability.

Acknowledgments

We thank the Aalto Science-IT project and CSC-IT Center for Science, Finland, for providing the computational resources, that were necessary to participate in this task. We would like to thank the shared task organizers for the excellent arrangements. We thank Feiyi Wang for the initial discussion.
References


Shervin Malmasi, Anjie Fang, Besnik Fetahu, Sudipta Kar, and Oleg Rokhlenko. 2022a. Multiconer: a large-scale multilingual dataset for complex named entity recognition.


