DAMO-NLP at SemEval-2022 Task 11:
A Knowledge-based System for Multilingual Named Entity Recognition

Xinyu Wang◇⋆, **Yongliang Shen**♠⋆, **Jiong Cai**◇⋆, **Tao Wang, Xiaobin Wang**†, **Pengjun Xie**†
Fei Huang†, **Weiming Lu**♠, **Yueting Zhuang**♠, **Kewei Tu**◇, **Wei Lu**‡, **Yong Jiang**†∗

†DAMO Academy, Alibaba Group
◇School of Information Science and Technology, ShanghaiTech University
♠College of Computer Science and Technology, Zhejiang University
‡StatNLP Research Group, Singapore University of Technology and Design
{wangxy1,caijiong,tukw}@shanghaitech.edu.cn
{syl,luwm}@zju.edu.cn, luwei@sutd.edu.sg
yongjiang.jy@alibaba-inc.com

Abstract

The MultiCoNER shared task aims at detecting semantically ambiguous and complex named entities in short and low-context settings for multiple languages. The lack of contexts makes the recognition of ambiguous named entities challenging. To alleviate this issue, our team **DAMO-NLP** proposes a knowledge-based system, where we build a multilingual knowledge base based on Wikipedia to provide related context information to the named entity recognition (NER) model. Given an input sentence, our system effectively retrieves related contexts from the knowledge base. The original input sentences are then augmented with such context information, allowing significantly better contextualized token representations to be captured. Our system wins 10 out of 13 tracks in the MultiCoNER shared task.[1]

1 Introduction

The MultiCoNER shared task ([Malmasi et al., 2022b](#)) aims at building Named Entity Recognition (NER) systems for 11 languages, including English, Spanish, Dutch, Russian, Turkish, Korean, Farsi, German, Chinese, Hindi, and Bangla. The task has three kinds of tracks including one multilingual track, 11 monolingual tracks and one code-mixed track. The multilingual track requires training multilingual NER models that are able to handle all languages. The monolingual tracks require training individual monolingual models where each model works for only one language. The code-mixed track requires handling code-mixed samples (sentences that may involve multiple languages). The datasets mainly contain sentences from three domains: Wikipedia, web questions and user queries,
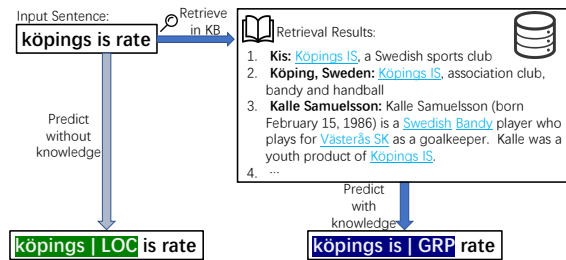

Figure 1: A motivating example from the English test set. In the retrieval results, the bold phrases are the title of the retrieved page and the underlined phrases contain the hyperlinks to other pages. LOC and GRP are entity labels representing location and group respectively.

which are usually short and low-context sentences. Moreover, these short sentences usually contain semantically ambiguous and complex entities, which makes the problem more difficult. In practice, professional annotators usually use their domain knowledge to disambiguate such kinds of entities. They may retrieve the related documents from a knowledge base (KB) or from a search engine to better guide them the annotation of ambiguous named entities ([Wang et al., 2019](#)). Therefore, we believe retrieving related knowledge can help the NER model to disambiguate hard samples in the shared task as well. A motivating example is shown in Figure 1, which shows how the retrieval results could help to improve the prediction in practice.

In this paper, we propose a general knowledge-based system for the MultiCoNER shared task. We propose to retrieve the related documents of the input sentence so that the recognition of difficult entities can be significantly eased. Based on Wikipedia of the 11 languages, we build a multilingual KB to search for the related documents of the input sentence. We then feed the input sentence and the related documents into the NER model. Moreover, we propose an iterative retrieval approach to im-

---

prove the retrieval quality. During training, we propose multi-stage fine-tuning. We first train a multilingual model so that the NER model can learn from all annotations. Next, we train the monolingual models (one for each language) and a code-mixed model by using the fine-tuned XLM-RoBERTa (XLM-R) (Conneau et al., 2020) embeddings in the multilingual model as initialization to further boost model performance on monolingual and code-mixed tracks. For each track, we train multiple models with different random seeds and use majority voting to form the final predictions.

Besides the system description, we make the following observations based on our experiments:

1. Knowledge-based systems can significantly improve both in- and out-of-domain performance compared with system without knowledge inputs.

2. Our multi-stage fine-tuning approach can help improve model performance in all the monolingual and code-mixed tracks. The approach can also reduce the training time to speed up our system building at different stages.

3. Our iterative retrieval strategy can further improve the retrieval quality and result in significant improvement on the performance of code-mixed track.

4. Searching over Wikipedia KB performs better than using online search engines on the Multi-CoNER datasets.

5. Comparing with other model variants we have tried, our NER model enjoys a good balance between model performance and speed.

2 Related Work

NER (Sundheim, 1995) is a fundamental task in natural language processing. The task has a lot of applications in various domains such as social media (Derczynski et al., 2017), news (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003), E-commerce (Fetahu et al., 2021; Wang et al., 2021b), and medical domains (Doğan et al., 2014; Li et al., 2016). Recently, pretrained contextual embeddings such as BERT (Devlin et al., 2019), XLM-R and LUKE (Yamada et al., 2020) have significantly improved the NER performance. The embeddings are trained on large-scale unlabeled data such as Wikipedia, which can significantly improve the contextual representations of named entities. Recent efforts (Peters et al., 2018; Akbik et al., 2018; Straková et al., 2019) concatenate different kinds

of pretrained embeddings to form stronger token representations. Moreover, the embeddings are trained over long documents, which allows the model to easily model long-range dependencies to disambiguate complex named entities in the sentence. Recently, a lot of work shows that utilizing the document-level contexts in the CoNLL NER datasets can significantly improve token representations and achieves state-of-the-art performance (Yu et al., 2020; Luoma and Pyysalo, 2020; Yamada et al., 2020; Wang et al., 2021a). However, the lack of context in the MultiCoNER datasets means the embeddings cannot take advantage of long-range dependencies for entity disambiguation. Recently, Wang et al. (2021b) use Google search to retrieve external contexts of the input sentence and successfully achieve state-of-the-art performance across multiple domains. We adopt this idea so that the embeddings can utilize the related knowledge by taking the advantage of long-range dependencies to form stronger token representations. Comparing with Wang et al. (2021b), we build the local KB based on Wikipedia because the KB matches the in-domain data of the shared task and is fast enough to meet the time requirement in the test phase[2].

Fine-tuning pretrained contextual embeddings is a useful and effective approach to many NLP tasks. Recently, some of the research efforts propose to further train the fine-tuned embeddings with specific training data or in a larger model architecture to improve model performance. Shi and Lee (2021) proposed two-stage fine-tuning, which first trains a general multilingual Enhanced Universal Dependency (Bouma et al., 2021) parser and then fine-tunes on each specific language separately. Wang et al. (2021a) proposed to train models through concatenating fine-tuned embeddings. We extend these ideas as multi-stage fine-tuning, which improves the accuracy of monolingual models that use fine-tuned multilingual embeddings as initialization in training. Moreover, multi-stage fine-tuning can accelerate the training process in system building.

3 Our System

We introduce how our knowledge-based NER system works in this section. Given a sentence of n tokens $\boldsymbol{x} = \{x_1, \cdots, x_n\}$, the sentence is fed into our knowledge retrieval module. The knowledge retrieval module takes the sentence as the query and retrieves top-k related paragraphs in KB.

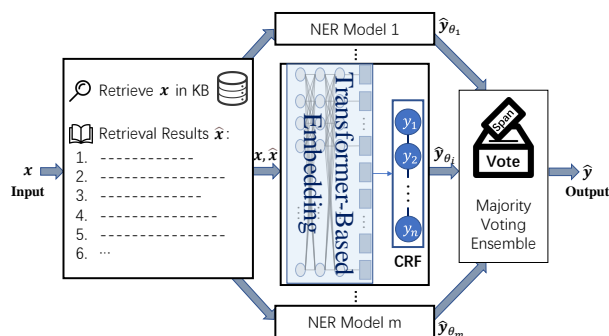[2]There are only 7 days for the test phase.

Figure 2: The architecture of our knowledge-based NER system.

The system then concatenates the input sentence and the related paragraphs together and feeds the concatenated sequence into the embeddings. The output token representations of the input sentence are fed into a linear-chain conditional random field (CRF) (Lafferty et al., 2001) layer and the CRF layer produces the label predictions. Given the label predictions of multiple NER models with different random seeds, the ensemble module uses a voting strategy to decide the final predictions $\hat{y} = \{\hat{y}_1, \cdots, \hat{y}_n\}$ of the sentence. The architecture of our framework is shown in Figure 2.

3.1 Knowledge Retrieval Module

Retrieval-augmented context is effective for named entity recognition tasks (Wang et al., 2021b), as external relevant contexts can provide auxiliary information for disambiguating complex named entities. We construct multilingual KBs based on Wikipedia pages of the 11 languages, and then retrieve relevant documents by using the input sentence as a query. These retrieved documents act as contexts and are fed into the NER module. To enhance the retrieval quality, we further designed an iterative retrieval approach, which incorporates predicted entities of NER models into the search query.

Knowledge Base Building Wikipedia is an evolving source of knowledge that can facilitate many NLP tasks (Chen et al., 2017; Verlinden et al., 2021). Wikipedia provides a rich collection of mention hyperlinks (referred to as wiki anchors). For example, in the sentence "*Steve Jobs founded Apple*", entities "*Steve Jobs*" and "*Apple*" are linked to the wiki entries Steve_Jobs and Apple_Inc respectively. For the NER task, these anchors provide useful clues on where the entities are to the model. Based on Wikipedia we can build local Wikipedia search engines to retrieve the relevant

context of the input sentences for each language.

We download the latest (2021-12-20) version of the Wikipedia dump from Wikimedia[3] and convert it to plain texts. Then we use ElasticSearch (ES)[4] to index them. ElasticSearch is document-oriented, and the document is the least searchable unit. We define the document in our local Wikipedia search engines with three fields: sentence, paragraph and title. We create inverted indexes on both the sentence field and the title field. The former is used as a sentence-level full-text retrieval field, while the latter indicates the core entity described by the wiki page and can be used as an entity-level retrieval field. The paragraph field stores the contexts of the sentence. To take advantage of the rich wiki anchors in Wikipedia paragraphs, we marked them with special markers. For example, to incorporate the hyperlinks [*Apple* → Apple Inc] and [*Steve Jobs* → Steve Jobs] to the paragraph, we transformed "*Steve Jobs founded Apple*" into "*<e:Steve Jobs>Steve Jobs</e> founded <e:Apple_inc>Apple</e>*"[5].

Sentence Retrieval Retrieval at the sentence level takes the input sentence as a query and retrieves the top-k documents on the sentence field. Given an input sentence, we select the corresponding search engine according to the language of the sentence.

Iterative Entity Retrieval The core of the NER task lies in the entities, while retrieval at the sentence level overlooks the key entities in the sentences. For this reason, we consider the relevance of the entities in the sentence to the title field in the documents during retrieval. We concatenate the entities in the sentences with "|" and then retrieve them on the title field. On the training and development sets, we utilize the ground-truth entities directly. On the test set, we first perform the sentence retrieval and then use the entity mentions[6] predicted by the model for entity retrieval. This bootstrapping manner can be applied for T turns.

Context Processing After top-k results from the KB are retrieved, the system post-processes the retrieved documents into the contexts of the input sentence. There are three options of utilizing

---

[3]https://dumps.wikimedia.org/
[4]https://www.elastic.co/
[5]<e:XXX>YYY</e>: where XXX is the title of the linked page and YYY is the phrase with hyperlink in the sentence.
[6]Here we define mentions as the named entities ignoring the entity types.

the texts in the documents, which are: 1) use the matched paragraph; 2) use the matched sentence; 3) use the matched sentence but remove the wiki anchors. We compare the performance of each option in section 5.4. In each retrieved document, we concatenate the title and texts together to form the context $\hat{x}_i$. The results are then concatenated into $\{\hat{x}_1, \cdots, \hat{x}_k\}$ based on the retrieval ranking.

3.2 Named Entity Recognition Module

In our system, we use XLM-R large as the embedding for all the tracks. It is a multilingual model and is applicable to all tracks. Given the input sentence x and the retrieved contexts $\{\hat{x}_1, \cdots, \hat{x}_k\}$, we add the separator token (i.e., "</s>" in XLM-R) between them and concatenated them together to form the input $\tilde{x}$ of the NER module. We chunk retrieved texts to avoid the amount of subtoken in the sequence exceeding the maximum subtoken length in XLM-R (i.e., 512 in XLM-R).

Our system regards the NER task as a sequence labeling problem. The embedding layer in the NER module encode the concatenated sequence $\tilde{x}$ and output the corresponding token representations $\{v_1, \cdots, v_n, \cdots\}$. The module then feeds the token representations $\{v_1, \cdots, v_n\}$ of the input sentence into a linear-chain CRF layer to obtain the conditional probability $p_\theta(y|\tilde{x})$:

$$\psi(y', y, v_i) = \exp(\mathbf{W}_y^T v_i + \mathbf{b}_{y',y}) \qquad (1)$$

$$p_\theta(y|\tilde{x}) = \frac{\prod_{i=1}^n \psi(y_{i-1}, y_i, v_i)}{\sum_{y' \in \mathcal{Y}(x)} \prod_{i=1}^n \psi(y'_{i-1}, y'_i, v_i)}$$

where θ represents the model parameters and $\mathcal{Y}(x)$ denotes the set of all possible label sequences given x. In the potential function $\psi(y', y, v_i)$, $\mathbf{W}_y^T v_i$ is the emission score and $\mathbf{b}_{y',y}$ is the transition score, where $\mathbf{W}^T \in \mathbb{R}^{t \times d}$ and $\mathbf{b} \in \mathbb{R}^{t \times t}$ are parameters and the subscripts y' and y are the indices of the matrices. During training, the negative log-likelihood loss $\mathcal{L}_{\text{NLL}}(\theta) = -\log p_\theta(y^*|\tilde{x})$ for the concatenated input sequence with gold labels y^* is used. During inference, the model prediction $\hat{y}_\theta$ is given by Viterbi decoding.

3.3 Ensemble Module

Given predictions $\{\hat{y}_{\theta_1}, \cdots, \hat{y}_{\theta_m}\}$ from m models with different random seeds, we use majority voting to generate the final prediction $\hat{y}$. We convert the label sequences into entity spans to perform majority voting. Following Yamada et al. (2020), the module ranks all spans in the predictions by the number of votes in descending order and selects the spans with more than 50% votes into the final prediction. The spans with more votes are kept if the selected spans have overlaps and the longer spans are kept if the spans have the same votes.

4 Experimental Setup

4.1 Data and Evaluation Methodology

We use the official MultiCoNER dataset (Malmasi et al., 2022a) in all tracks to train our NER models. There are mainly three domains in the dataset: LOWNER (Low-Context Wikipedia NER) contains low-context sentences from Wikipedia; MSQ (MS-MARCO Question NER) is based on MS-MARCO web question corpus (Nguyen et al., 2016) containing a lot of natural language questions; ORCAS (Search Query NER) contains user queries from Microsoft Bing (Craswell et al., 2020). The MSQ and ORCAS samples are taken as out-of-domain data in the shared task. The training and development sets only contain a small collection of samples of these two domains and mainly contain data from the LOWNER domain. The test set, however, contains much more MSQ and ORCAS samples to assess the out-of-domain performance.

The results of the shared task are evaluated with the entity-level macro F1 scores, which treat all the labels equally. In comparison, most of the publicly available NER datasets (e.g., CoNLL 2002, 2003 datasets) are evaluated with the entity-level micro F1 scores, which emphasize common labels.

4.2 Training

NER Model Training   Before building the final system, we compare a lot of variants of the system. We train these variant models on the training set for 3 times each with different random seeds and compare the averaged performance of the models. According to the dataset sizes, we train the models for 5 epochs, 10 epochs and 100 epochs for multilingual, monolingual and code-mixed models respectively. Our final NER models are trained on the combined dataset including both the training and development sets on each track to fully utilize the labeled data. For models trained on the training set, we use the best macro F1 on the development set during training to select the best model checkpoint. For models trained on the combined dataset,

System	EN	ES	NL	RU	TR	KO	FA	DE	ZH	HI	BN	MULTI	MIX	AVG.
Ours: Baseline	77.81	76.80	80.51	74.65	72.83	70.81	72.68	81.92	65.56	67.80	65.27	74.19	77.75	73.74
Sliced	74.54	75.11	77.66	73.73	68.77	70.66	68.66	78.90	65.21	67.00	63.05	71.07	72.74	71.32
RACAI	75.78	75.62	78.41	74.60	70.42	71.74	70.42	79.39	62.70	68.08	66.28	72.10	79.37	72.69
USTC-NELSLIP	85.47	85.44	87.67	83.82	85.52	86.36	87.05	89.05	**81.69**	84.64	**84.24**	85.30	**92.90**	86.09
Ours	**91.22**	**89.94**	**90.50**	**91.50**	**88.69**	**88.59**	**89.70**	**90.65**	78.06	**86.23**	83.51	**85.31**	91.79	**88.13**

Table 1: Part of the official results and the post-evaluation results of our baseline system.

we use the final model checkpoint after training[7].

Multi-stage Fine-tuning Besides our final settings, we have a lot of stages of KB settings during our system building. Multi-stage fine-tuning aims at transferring the parameters of fine-tuned embeddings in a model at an early stage into other models in the next stage. The approach stores the checkpoint of fine-tuned XLM-R embeddings at the early stage and uses it as the initialization of XLM-R embeddings for model training at the next stage. One benefit of multi-stage fine-tuning is the monolingual and code-mixed models, can utilized the annotations of all the tracks to further improve the model performance. XLM-R embedding is a multilingual embedding with strong cross-lingual transferability over all 11 languages. Therefore, we use the checkpoint of fine-tuned multilingual model for continue fine-tuning on the monolingual and code-mixed models. Another benefit of multi-stage fine-tuning is that it accelerates the training speed. As the size of the multilingual dataset is relatively large, it is quite time-consuming to train a multilingual model. When we try different types of KB, we can utilize the checkpoints of multilingual models at the previous stage to train the monolingual and code-mixed models with new types of contexts without training new multilingual models. Moreover, we can reduce the training epochs for faster speed since the XLM-R checkpoints have already learned from all the datasets.

Continue Pretraining To make XLM-R learn the data distribution of the shared task, we combine the training and development sets on the monolingual tracks to build a corpus to continue pretrain XLM-R. Specifically, we collocate all sentences according to their languages, then cut the text into chunks of fixed length, and train the model on these text chunks using the Masked Language Modeling objective. We continue pretrain XLM-R for 5 epochs. We use the continue pretrained XLM-R model as the initialization of the multilingual

models during training.

5 Results and Analysis

In this section, we use language codes[8] to represent languages, and use **MULTI** and **MIX** to represent multilingual and code-mixed tracks respectively[9].

5.1 Main Results

There are 55 teams that participated in the shared task. Due to limited space, we only compare our system with the systems from teams USTC-NELSLIP, RACAI and Sliced[10]. In the post-evaluation phase, we evaluate a baseline system without using the knowledge retrieval module to further show the effectiveness of our knowledge-based system. The official results and the results of our baseline system are shown in Table 1. Our system performs the best on 10 out of 13 tracks and is competitive on the other 3 tracks. Moreover, our system outperforms our baseline by 14.39 F1 on average, which shows the knowledge retrieval module is extremely helpful for disambiguating complex entities leading to significant improvement on model performance.

5.2 How Significant is the Role of Knowledge-based System on Each Domain?

To further show the effectiveness of our knowledge-based system, we show the relative improvements of our system over our baseline system on each domain in Table 2. We observe that in most of the cases, the two out-of-domain test sets have more relative improvements than the in-domain test set. This observation shows that the knowledge from Wikipedia can not only improve the performance of the LOWNER domain which is the same domain as the KB, but also has very strong cross-domain

---

[7]Please refer to Appendix A for detailed settings.

[8]https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes
[9]Please refer to Appendix B for more analysis.
[10]Please refer to https://multiconer.github.io/results for more details about the results.

			EN	ES	NL	RU	TR	KO	FA	DE	ZH	HI	BN	AVG.
In-domain	LOWNER	Baseline	88.70	86.54	89.92	81.52	88.52	86.25	81.85	91.71	85.43	83.13	82.69	86.02
		Ours	96.78	96.19	97.96	96.60	96.43	96.83	96.48	94.89	88.66	84.18	86.31	93.76
		Δ	+8.08	+9.65	+8.04	+15.08	+7.91	+10.58	+14.63	+3.18	+3.23	+1.05	+3.62	+7.74
Out-of-domain	MSQ	Baseline	70.49	71.86	72.63	72.31	75.49	68.57	71.54	74.63	67.38	73.57	58.66	70.65
		Ours	83.50	83.10	83.34	87.03	88.76	81.96	87.36	86.18	79.80	89.20	72.00	83.84
		Δ	+13.01	+11.24	+10.71	+14.72	+13.27	+13.39	+15.82	+11.55	+12.42	+15.63	+13.34	+13.19
	ORCAS	Baseline	62.07	62.71	67.39	64.83	66.92	56.08	65.52	67.52	55.34	62.03	60.68	62.83
		Ours	83.72	81.33	80.29	85.00	85.85	81.06	84.84	84.40	72.11	85.75	82.13	82.41
		Δ	+21.65	+18.62	+12.90	+20.17	+18.93	+24.98	+19.32	+16.88	+16.77	+23.72	+21.45	+19.58

Table 2: Per-domain macro F1 score on the test set of our system and our baseline system for each language. Δ represents the relative improvements of our system over the baseline system.

transferability to other domains such as web questions and user queries. According to the baseline performance over the three domains, the ORCAS domain has the lowest score, which shows the challenges in recognizing named entities in user queries. However, our retrieved documents in KB can significantly ease the challenges in this domain and results in the highest improvement out of the three domains.

5.3 How Relevant Are the Retrieval Results to the Queries?

To evaluate the relevance of the retrieval results to the query, we define a character-level relevance metric, which calculates the Intersection-over-Union (IoU) between the characters of query and result. Assuming that the character sets[11] of query and retrieval result are A and B respectively, then the character-level IoU is $\frac{A \cap B}{A \cup B}$. We calculate the character-level IoU of the sentence and its top-1 retrieval result on all tracks, and plot its distribution on the training, development and test set in Figure 3. We have the following observations: 1) the IoU values are concentrated around 1.0 on the training and development sets of **EN**, **ES**, **NL**, **RU**, **TR**, **KO**, **FA**, which indicates that most of the samples were derived from Wikipedia. Therefore, by retrieving, we can obtain the original documents for these samples. 2) the distribution of data on the test set is consistent with the training and development sets for most languages, except for **TR**. On **TR**, the character-level IoU values of the samples and query results cluster at around 0.5. We hypothesize that this is because the source of the test set for **TR** is different from the training set. However, the model still performs strongly on this language, suggesting that the model can mitigate the difficulties caused by inconsistent data distribution by retrieving the context from Wikipedia.

5.4 How Important Can the Types of KB be?

We compare several types of KBs and contexts during our system building.

Online Search Engine In the early stage, we tried to use the knowledge retrieved from Google Search, which can retrieve related knowledge from a large scale of webs and is believed to be a strong multilingual search engine.

Three Context Types Retrieved from Wikipedia As we mentioned in Section 3.1, there are three context processing options, which are: 1) use the matched paragraph; 2) use the matched sentence; 3) use the matched sentence but remove the wiki anchors. We denote the three options as PARA, SENT and SENT-LINK respectively.

Entity Retrieval with Gold Entities We use gold entities on the development set to see whether the model performance can be improved. This can be seen as the most ideal scenario for iterative retrieval. We denote this process as ITER$_G$ and use PARA for the context type.

In Table 3, we can observe that: 1) For the three context options, PARA is the best option for **EN**, **ES**, **NL**, **RU**, **TR**, **KO**, **FA**, **MIX** and **MULTI**. SENT-LINK is the best option for **HI** and **BN**. For **DE** and **ZH**, SENT and SENT-LINK are competitive. As a result, we choose SENT for the two languages since we believe the wiki anchors from the Wikipedia can help model performance; 2) Comparing with the baseline, the knowledge from Google Search can improve model performance. Based on the best context option of each track, the knowledge from Wikipedia is better than the online search engine; 3) For ITER$_G$, we can find that the context can further

---

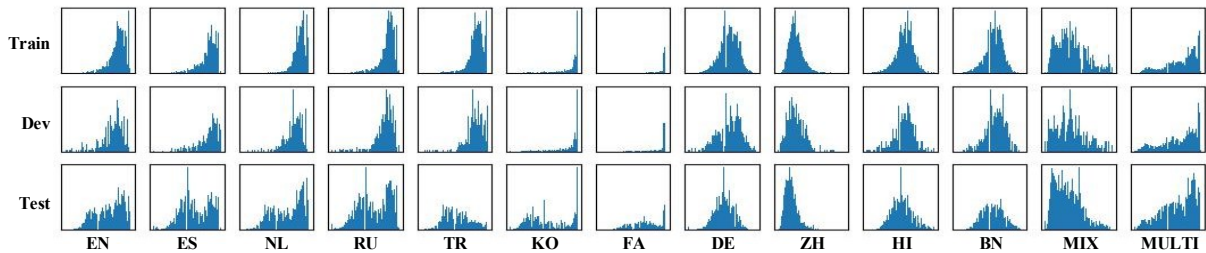[11] The sets take repeat characters as different characters.

Figure 3: The distribution of the character-level IoU between the query and its top-1 result. Each subplot is a histogram on the corresponding dataset, where the *x*-axis indicates the IoU values ranging from 0 to 1.

	EN	ES	NL	RU	TR	KO	FA	DE	ZH	HI	BN	MULTI	MIX
Baseline	87.13	85.88	88.87	82.38	86.22	85.98	81.25	91.21	87.65	82.62	82.80	85.78	77.92
Google Search	92.46	88.68	91.58	85.88	89.83	88.95	82.96	93.56	89.16	84.27	84.38	87.84	86.26
Wiki-PARA	**95.82**	94.19	**97.53**	95.53	**97.40**	96.05	**95.93**	92.83	87.10	82.78	83.35	93.51	85.16
Wiki-SENT	87.62	89.33	92.90	79.41	89.00	91.49	95.99	94.42	**89.47**	84.55	84.12	89.34	78.65
Wiki-SENT$_{-LINK}$	86.83	87.65	91.86	79.15	86.66	86.36	84.37	94.46	89.32	84.78	84.83	87.35	80.07
Wiki-PARA+ITER$_G$	94.89	**94.44**	97.45	**95.59**	96.89	**96.34**	95.83	**94.62**	88.47	**86.43**	**85.85**	**93.60**	**90.52**

Table 3: A comparison of the models utilizing different types of knowledge on the **development set**.

	HI	BN	MIX
Wiki-PARA+ITER$_G$	86.43	85.85	90.52
Wiki-SENT+ITER$_G$	85.69	86.57	91.38
Wiki-SENT$_{-LINK}$+ITER$_G$	86.15	86.13	91.38
Wiki-Opt$_{Best}$	**84.78**	**84.83**	85.16
Wiki-Opt$_{Best}$+ITER$_P$	83.36	84.37	**88.97**

Table 4: Effectiveness of iterative retrieval. Opt$_{Best}$ represents using the best context option for each language.

	HI	BN	MIX
Wiki-Opt$_{Best}$	90.02	90.81	96.72
Wiki-Opt$_{Best}$-Mention	90.76	90.75	96.71

Table 5: A comparison of mention detection F1 score over NER models and mention detection models.

Module	Sentences/Second
Local Knowledge Base Retrieval	64.52
Google Search Retrieval	1.50
NER Module - Training	2.91
NER Module - Prediction	8.13

Table 6: Model speed of the knowledge retrieval module and NER module in our system.

improve the performance over 8 out of 13 tracks. However, there are only significant improvements for **HI**, **BN** and **MIX**.

Iterative Entity Retrieval with Predicted Entities Based on the results in Table 3, we further analyze how the predicted entity mentions can improve the retrieval quality. We denote the iterative

entity retrieval with predicted mentions as ITER$_P$. In the experiment, we set $T = 2$.[12] We extract the predicted mentions of the development sets from the models based on the best context option for each track. We conduct the experiments over **HI**, **BN** and **MIX** which have significant improvement with ITER$_G$. In Table 4, we also list the performance of ITER$_G$ for reference, which can be seen as using the predicted mentions with 100% accuracy. From the results, we observe that only **MIX** can be improved.

Since iterative entity retrieval uses predicted mentions as a part of retrieval query, the performance of mention detection directly affects the retrieval quality. To further analyze the observation in Table 4, we evaluate the mention F1 score of the NER models with sentence retrieval. For comparison with mention detection performance of NER models, we additionally train mention detection models by discarding the entity labels during training. From the results in Table 5, we suspect the low mention F1 introduces noises in the knowledge retrieval module for **BN** and **HI**, which lead to the decline of performance as shown in Table 4. Moreover, the mention F1 of mention detection models (second row of Table 5) only outperform that of the NER models (first row of Table 5) in a moderate scale. Therefore, we train the ITER models only for the code-mixed track and use the NER models with sentence retrieval to predict mentions.

---

[12]Our preliminary experiments show that there is no significant improvement for $T = 3$.

	EN	ES	NL	RU	TR	KO	FA	DE	ZH	HI	BN	MIX	AVG.
XLM-R	92.46	88.68	91.58	85.88	89.83	88.95	82.96	93.56	89.16	84.27	84.38	84.52	88.02
CE	**92.49**	**88.97**	**92.20**	**86.21**	**90.47**	**89.01**	**83.53**	**93.96**	**89.40**	**84.86**	**85.38**	**87.35**	**88.65**

Table 7: A comparison of CE models and XLM-R models. Both kinds of models utilize the knowledge from Google Search. The scores are the averaged macro F1 score on the **development set**.

	EN	ES	NL	RU	TR	KO	FA	DE	ZH	HI	BN	MIX	AVG.
Baseline w/ MF	**87.13**	**85.88**	**88.87**	**82.38**	**86.22**	**85.98**	**81.25**	**91.21**	**87.65**	**82.62**	**82.80**	**77.92**	**84.99**
Baseline w/o MF	85.88	84.28	87.98	81.01	84.61	83.98	79.98	89.54	85.57	79.90	81.18	68.21	82.68

Table 8: A comparison of training the NER models with and without multi-stage fine-tuning (MF) for our baseline system on the **development set**.

	EN	ES	NL	RU	TR	KO	FA	AVG.
XLM-R	95.82	94.19	97.53	95.53	97.40	96.05	95.93	96.07
Ensem	96.56	95.11	97.83	**96.48**	97.57	96.54	96.15	96.61
ACE	**96.69**	**95.80**	**98.22**	96.46	**98.01**	**96.79**	**96.75**	**96.96**

Table 9: A comparison of ACE models, XLM-R models and an ensemble of the XLM-R models on the **development set**.

5.5 Model Efficiency

Table 6 shows the speed of each module in our system. In the table, we also show that the retrieval speed of our local KB is significantly faster than that of Google Search. The bottleneck of the system speed is the NER module rather than the knowledge retrieval module. The main reason for the slow speed of the NER module is that the input length of the knowledge-based system is significantly longer than the original input. Taking the EN test set as an example, there are on average 10 tokens for each input sentence in the original test set while there are 218 tokens for the input of our knowledge-based system. The longer inputs slow down the encoding at XLM-R embeddings.

5.6 Effect of Embedding Concatenation

We compare with some variants of our system that we designed but did not use in the test phase.

CE (Concatenation of Embeddings) CE is one of the usual approaches to NER, which concatenates different kinds of embeddings to improve the token representations. In the early stage of our system building, we compare CE with only using the XLM-R embeddings based on the knowledge retrieved from the Google Search. Results in Table 7 show that CE models are stronger than the models using XLM-R embeddings only in all the cases, which show the effectiveness of CE.

ACE (Automated Concatenation of Embeddings) ACE (Wang et al., 2021a) is an improved version of CE which automatically selects a better concatenation of the embeddings. We use the same embedding types as CE and the knowledge are from our Wikipedia KB. We experiment on EN, ES, NL, RU, TR, KO and FA, which are strong with PARA contexts. In Table 9, we further compare ACE with ensemble XLM-R models. Results show ACE can improve the model performance and even outperform the ensemble models[13].

The results in Table 7 and 9 show the advantage of the embedding concatenation. However, as we have shown in Section 5.5, the prediction speed is quite slow with the single XLM-R embeddings. The CE models further slow down the prediction speed since the models contain more embeddings. The ACE models usually have faster prediction speed than the CE models. However, training the ACE models is quite slow. It takes about four days to train a single ACE model. Moreover, the ACE models cannot use the development set to train the model since they use development score as the reward to select the embedding concatenations. Therefore, due to the time constraints, we did not use these two variants in our submission during the shared task period.

5.7 Effectiveness of Multi-stage Fine-tuning

In Table 8, we show the effectiveness of multi-stage fine-tuning on the development set for our baseline system. The result shows that multi-stage fine-tuning can significantly improve the model performance for all the tracks.

---

[13]Please refer to Appendix A.3 for detailed settings.

6 Conclusion

In this paper, we describe our knowledge-based system for the MultiCoNER shared task, which wins 10 out of 13 tracks in the shared task. We construct multilingual KBs and retrieve the related documents from KBs to enhance the token representations of input text. We show that the NER models can use the retrieved knowledge to facilitate complex entity prediction, significantly improving both the in-domain and out-of-domain performance. Multi-stage fine-tuning can help the monolingual models learn from the training data of all the languages and improve the model performance and training efficiency. We also show that the system presents a good balance between the model performance and prediction efficiency to meet the time requirement in the test phase. We believe this system can be widely applied to other domains for the task of NER. For future work, we plan to improve the retrieval quality and adopt the system to support other kinds of entity-related tasks.

Acknowledgements

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Gosse Bouma, Djamé Seddah, and Daniel Zeman. 2021. From raw text to enhanced Universal Dependencies: The parsing shared task at IWPT 2021. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 146–157, Online. Association for Computational Linguistics.

Wanxiang Che, Yijia Liu, Yuxuan Wang, Bo Zheng, and Ting Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 55–64, Brussels, Belgium. Association for Computational Linguistics.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Nick Craswell, Daniel Campos, Bhaskar Mitra, Emine Yilmaz, and Bodo Billerbeck. 2020. Orcas: 20 million clicked query-document pairs for analyzing search. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2983–2989.

Leon Derczynski, Eric Nichols, Marieke van Erp, and Nut Limsopatham. 2017. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 140–147, Copenhagen, Denmark. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.

Besnik Fetahu, Anjie Fang, Oleg Rokhlenko, and Shervin Malmasi. 2021. Gazetteer enhanced named entity recognition for code-mixed web queries. In *SIGIR '21*, SIGIR '21, New York, NY, USA. Association for Computing Machinery.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wiegers, and

Zhiyong Lu. 2016. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database: The Journal of Biological Databases and Curation*, 2016.

Jouni Luoma and Sampo Pyysalo. 2020. Exploring cross-sentence contexts for named entity recognition with BERT. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 904–914, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Shervin Malmasi, Anjie Fang, Besnik Fetahu, Sudipta Kar, and Oleg Rokhlenko. 2022a. MultiCoNER: a Large-scale Multilingual dataset for Complex Named Entity Recognition.

Shervin Malmasi, Anjie Fang, Besnik Fetahu, Sudipta Kar, and Oleg Rokhlenko. 2022b. SemEval-2022 Task 11: Multilingual Complex Named Entity Recognition (MultiCoNER). In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*. Association for Computational Linguistics.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. In *CoCo@ NIPS*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Tianze Shi and Lillian Lee. 2021. TGIF: Tree-graph integrated-format parser for enhanced UD with two-stage generic- to individual-language finetuning. In *Proceedings of the 17th International Conference on Parsing Technologies and the IWPT 2021 Shared Task on Parsing into Enhanced Universal Dependencies (IWPT 2021)*, pages 213–224, Online. Association for Computational Linguistics.

Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural architectures for nested NER through linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331, Florence, Italy. Association for Computational Linguistics.

Beth M. Sundheim. 1995. Named entity task definition, version 2.1. In *Proceedings of the Sixth Message Understanding Conference*, pages 319–332.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. 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.

Severine Verlinden, Klim Zaporojets, Johannes Deleu, Thomas Demeester, and Chris Develder. 2021. Injecting knowledge base information into end-to-end joint entity and relation extraction and coreference resolution. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1952–1957, Online. Association for Computational Linguistics.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021a. Automated Concatenation of Embeddings for Structured Prediction. In *the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021)*. Association for Computational Linguistics.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021b. Improving named entity recognition by external context retrieving and cooperative learning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1800–1812, Online. Association for Computational Linguistics.

Zihan Wang, Jingbo Shang, Liyuan Liu, Lihao Lu, Jiacheng Liu, and Jiawei Han. 2019. CrossWeigh: Training named entity tagger from imperfect annotations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5154–5163, Hong Kong, China. Association for Computational Linguistics.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online. Association for Computational Linguistics.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6470–6476, Online. Association for Computational Linguistics.

A   Detailed Experimental Setup

The detailed statistics of the MultiCoNER dataset are listed in Table 10 and the statistics of our KBs ares shown in Table 11.

A.1 Statistics of Datasets and Knowledge Bases

Track	Train	Dev	Test
English	15,300	800	217,818
Spanish	15,300	800	217,887
Dutch	15,300	800	217,337
Russian	15,300	800	217,501
Turkish	15,300	800	136,935
Korean	15,300	800	178,249
Farsi	15,300	800	165,702
German	15,300	800	217,824
Chinese	15,300	800	151,661
Hindi	15,300	800	141,565
Bangla	15,300	800	133,119
Multilingual	168,300	8,800	471,911
Code-mixed	1,500	500	100,000

Table 10: Statistics of the the MultiCoNER dataset (# of sentences). Note that the training and development sets of the multilingual dataset are a mixture of monolingual training and development sets respectively.

Language	Pages	Paragraphs	ES Docs
English	8,075,229	138,259,937	224,077,884
Spanish	1,813,109	29,767,543	47,248,391
Dutch	2,234,442	18,007,520	29,442,016
Russian	2,437,595	44,536,255	77,903,362
Turkish	728,950	8,196,825	12,685,674
Korean	905,976	11,965,418	16,326,787
Farsi	1,502,301	13,723,218	17,342,825
German	3,147,933	54,315,261	98,386,199
Chinese	1,659,253	20,342,685	14,888,964
Hindi	196,745	1,926,636	3,279,827
Bangla	203,869	2,526,333	4,342,959

Table 11: Detailed statistics on 11 languages.

A.2 System Configurations

For the knowledge retrieval module, we retrieve top-10 related results from the KB. For iterative entity retrieval, we set $T = 2$. In masked language model pretraining, we use a learning rate of 5×10^{-5}. For the NER module, we use a learning rate of 5×10^{-6} for fine-tuning the XLM-R embeddings and use a learning rate of 0.05 to update the parameters in the CRF layer following Wang et al. (2021b). Each NER model built by our system can be trained and evaluated on a single Tesla V100 GPU with 16GB memory. For the ensemble module, we train about 10 models for each track.

A.3 Settings of CE and ACE models

In Section 5.6, we compare our NER model with CE and ACE models. In CE and ACE models, we concatenate monolingual fastText (Bo-

	DE	ZH	HI	BN	MIX	AVG.
Voting	**94.65**	**89.18**	**85.51**	**85.22**	**86.57**	**88.23**
CRF	94.04	88.96	85.37	85.12	85.33	87.76

Table 12: A comparison of ensemble approaches on the **development set**.

janowski et al., 2017) word embeddings, monolingual/multilingual Flair embeddings (Akbik et al., 2018), ELMo embeddings (Peters et al., 2018; Che et al., 2018), XLM-R embeddings fine-tuned on the whole training data and XLM-R embeddings fine-tuned on the language data by multi-stage fine-tuning. We only feed the knowledge-based input into XLM-R embeddings and feed the original input into other embeddings because it is hard for the other embeddings (especially for LSTM-based embeddings such as Flair and ELMo) to encode such a long input. We use Bi-LSTM encoder to encode the concatenated embeddings with a hidden state of 1,000 and then feed the output token representations into the CRF layer. Following most of the previous efforts, we use SGD optimizer with a learning rate of 0.01. For ACE, we search the embedding concatenation for 30 episodes.

B More Analysis

B.1 Majority Voting Ensemble and CRF Level Ensemble

As we state in Section 3.3, we use majority voting as the ensemble algorithm in our system. We show an experiment about how the voting threshold affect the ensemble model performance during our system building on the development set. We ensemble the models on **DE**, **ZH**, **HI**, **BN**, **MIX** with PARA since these five tracks have relatively lower performance than the other 7 tracks. In Figure 4, we show how the threshold of the majority voting affects the model performance. From the figure, we can see that the best threshold varies over the language. Therefore, we simply choose 0.5 as there is no best threshold value. Moreover, we compare the majority voting ensemble and CRF level ensemble in Table 12. The CRF level ensemble averages the emission and transition scores in the Eq. 1 predicted by the candidate models and uses the Viterbi algorithm to get the prediction. The results show that CRF level ensemble performs inferior to the majority voting ensemble. The possible reason is that training with different random seeds may lead to different emission transition scores at different
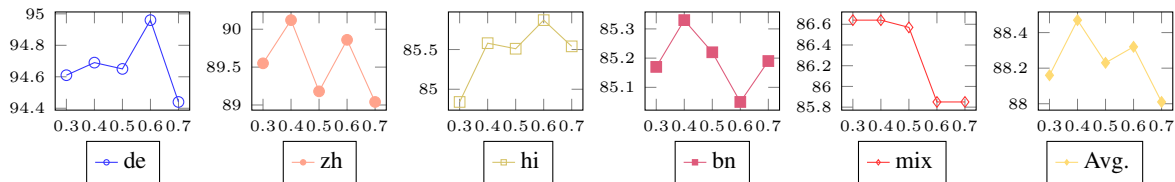
Figure 4: An illustration of majority voting threshold versus the ensemble model performance.

| Test Context | PARA | | Opt$_{Best}$ | |
Search KB	All	Language	All	Language
Wiki-PARA	84.57	84.94	-	-
Wiki+Opt$_{Best}$	-	84.96	84.38	84.78

Table 13: Test results for multilingual models with different context options and different KB size.

scales. As a result, the models with larger scales have higher weights in the ensemble.

B.2 How the Search Space and the Context Type Affects Multilingual Model Performance?

In the multilingual test set, we can find 304,905 sentences in the other monolingual test sets while there are 167,006 sentences that cannot be found. For these sentences, we can either search on the whole KB of all languages or first detect the language of the input sentence and then search in the specific language KB[14]. Moreover, as we discussed in Section 5.4, using different kinds of retrieved knowledge affects the model performance. As a result, we train two types of multilingual models. One is only using the PARA contexts for all language and another is using the best option for each language based on Table 3. From the results in Table 13, we can observe that: 1) searching over the language specific KB performs better than searching the whole KB, 2) using the language specific context option cannot improve the model performance. Therefore, we ensemble both types of the model for the final submission.

---

[14]We determine the language of the input sentence using the langdetector (https://pypi.org/project/langdetect/) tool.