MIA 2022 Shared Task Submission: Leveraging Entity Representations, Dense-Sparse Hybrids, and Fusion-in-Decoder for Cross-Lingual Question Answering

Zhucheng Tu and Sarguna Janani Padmanabhan
Apple
{zhucheng_tu,jananip}@apple.com

Abstract

We describe our two-stage system for the Multilingual Information Access (MIA) 2022 Shared Task on Cross-Lingual Open-Retrieval Question Answering. The first stage consists of multilingual passage retrieval with a hybrid dense and sparse retrieval strategy. The second stage consists of a reader which outputs the answer from the top passages returned by the first stage. We show the efficacy of using entity representations, sparse retrieval signals to help dense retrieval, and Fusion-in-Decoder. On the development set, we obtain 43.46 F1 on XOR-TyDi QA and 21.99 F1 on MKQA, for an average F1 score of 32.73. On the test set, we obtain 40.93 F1 on XOR-TyDi QA and 22.29 F1 on MKQA, for an average F1 score of 31.61. We improve over the official baseline by over 4 F1 points on both the development and test sets.[1]

1 Introduction

This paper describes our submission to the Multilingual Information Access (MIA) 2022 Shared Task on Cross-Lingual Open-Retrieval Question Answering. Cross-lingual open-retrieval question answering is the task of finding an answer to a knowledge-seeking question in the same language as the question from a collection of documents in many languages. The answer may not necessarily exist in a document that's in the same language as the question, and hence a system need to find the answer across relevant documents in a different language. The shared task at Multilingual Information Access 2022 evaluates cross-lingual open-retrieval question answering systems using two datasets, XOR-TyDi QA (Asai et al., 2020) and MKQA (Longpre et al., 2020).[2]

We use a two stage approach, similar to the CORA (Asai et al., 2021) baseline, where the first stage performs multilingual passage retrieval and the second stage performs cross-lingual answer generation. In the first stage, we leverage mLUKE (Ri et al., 2021), a pretrained language model that models entities, to train a dual encoder that encodes the question and passage separately (Karpukhin et al., 2020). During retrieval, we perform nearest neighbor search using the query vector on an index of encoded passage vectors. We merge these dense retrieval hits with BM25 sparse retrieval hits using an algorithm we call Sparse-Corroborate-Dense. Finally, we feed the ranked list of passages into a reader based on Fusion-in-Decoder (Ri et al., 2021) and mT5 (Xue et al., 2020) to produce the final answer. We do not perform iterative training to repeat these steps multiple times.

Compared to official baseline 1, we improve the macro-averaged score by 4.1 F1 points. We perform analysis to show the effectiveness of entity representations, using sparse signals to improve dense hits, and Fusion-in-Decoder.

2 Data and Processing

2.1 Datasets

We use the official training data consisting of 76635 English questions and answers from Natural Questions (Kwiatkowski et al., 2019) and 61360 questions and answers from XOR-TyDi QA (Asai et al., 2020) to train our dual encoder model. We do not train on the development data or the subsets of the Natural Questions and TyDi QA (Clark et al., 2020) data, which are used to create MKQA or XOR-TyDi QA data. For training the reader, we leverage Wikipedia language links, which is detailed in Section 3.3.

XOR-TyDi QA consists of annotated questions and short answers across seven typologically diverse languages. It can be broken down into two subsets, questions where the answer can be found in a passage in the same language as the question

[1] Our submission team name is Team Utah: https://eval.ai/challenge/1638/leaderboard/3933.

[2] https://mia-workshop.github.io/shared_task.html.

("in-language"), which just come from answerable questions in TyDi QA (Karpukhin et al., 2020), and questions where the answer is unanswerable from a passage in the same language as the question and can only be found in an English passage ("cross-lingual"), which are newly added answers in XOR-TyDi QA. A system should be able to succeed at both monolingual retrieval and cross-lingual retrieval.

MKQA (Longpre et al., 2020) consists of 10K parallel questions and answers across 26 typologically diverse locales. The original question is taken from Natural Questions (Kwiatkowski et al., 2019) in English and translated to 25 different locales. MKQA does not contain any data for training and is only used for evaluation.

2.2 Passage Corpus

We directly use the passages corpus provided by the shared task, with the addition of Tamil (`ta`) and Tagalog (`tl`) which are not included in the baseline's passage data. Following the other languages, we use the `20190201` snapshot of the Wikipedia dumps. We follow the same preprocessing steps as the baseline passages data.[3] We manually split the data into language-specific files, which are later used to build language-specific dense and sparse indices. Final passage retrieval results are aggregated among different indices. The number of passages in each language is shown in Table 1.

Language	Passages	% of Total Passages
Arabic (`ar`)	1304828	2.83
Bengali (`bn`)	179936	0.39
English (`en`)	18003200	39.00
Spanish (`es`)	5738484	12.43
Finnish (`fi`)	886595	1.92
Japanese (`ja`)	5116905	11.09
Khmer (`km`)	63037	0.14
Korean (`ko`)	638865	1.38
Malaysian (`ms`)	397396	0.86
Russian (`ru`)	4545634	9.85
Swedish (`sv`)	4525695	9.81
Tamil (`ta`)	219356	0.48
Telugu (`te`)	274230	0.59
Tagalog (`tl`)	69228	0.15
Turkish (`tr`)	798368	1.73
Chinese (`zh`)	3394943	7.36
Total	**46156700**	**100.0**

Table 1: Number of passages in corpus for each language.

3 System Architecture and Pipeline

Our system differs from the baseline in three ways. First, in the passage retrieval step, we replace mBERT (Devlin et al., 2019) with mLUKE (Ri et al., 2021). Second, we construct sparse indices from which we will retrieve passages to augment dense retriever-retrieved passages, inspired by Zhang et al. (2021) but uses a different dense-sparse hybrid approach. Finally, we encode each question and passage independently as opposed to all passages together following the Fusion-in-Decoder (Izacard and Grave, 2020) approach.

3.1 Entity Representations

For dense retrieval, we use a multilingual pretrained language model with entity representations, mLUKE (Ri et al., 2021), to initialize the dual encoder in DPR (Karpukhin et al., 2020). We use the same training objective as DPR and also use the last layer's hidden state of the first input token as the representation for both the question and passage. mLUKE is a multilingual extension of LUKE (Yamada et al., 2020), a pretrained contextualized representation of words and entities based on the Transformer (Vaswani et al., 2017). Words and entities are treated as different types of tokens and the entity-aware self-attention mechanism leads to improved effectiveness. We use the Hugging Face transformers (Wolf et al., 2020) versions of `mluke-base` and `bert-base-multilingual-uncased`. Only in-batch negatives are used to train the dual encoder.

3.2 Dense-Sparse Hybrids

In order to effectively retrieve passages in a multilingual setting, the retrieval component needs to do well in both monolingual retrieval and cross-lingual retrieval. Monolingual retrieval is the setting where we want to retrieve passages in the same language as the question. For more than half of the questions in the XOR-TyDi QA dataset, for example, the answer is found in a passage that's in the same language as the question. Cross-lingual retrieval is the setting where we want to retrieve relevant passages in different language from the question. We use both sparse retrieval (i.e. BM25) and dense

[3] https://github.com/mia-workshop/MIA-Shared-Task-2022/commits/main/baseline/wikipedia_preprocess/build_dpr_w100_data.py

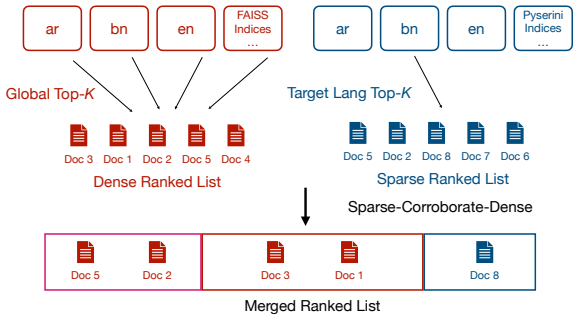

Figure 1: An illustration of the Sparse-Dense-Corroborate algorithm, running with $K = 5$ and $max\_frac = 0.6$ for a Bengali (bn) question. We first retrieve the 5 passages with the highest scores from the dense indices, and the top 5 passages from the Bengali sparse index. For the first output slice, we take passages in both lists, ordered by same order as in the dense list, which are doc 5 and doc 2. For the second slice, we add the top remaining passages from the dense list, which are doc 3 and doc 1. For the third slice, we take the top remaining passages from the sparse list, which is doc 8. The max number of sparse results that is allowed to influence the final list is $0.6 \times 5 = 3$, which are docs 5, 2, and 8.

retrieval together in our system. Experiments in Mr. TyDi (Zhang et al., 2021) indicate BM25 outperforms DPR (Karpukhin et al., 2020) for the languages in XOR-TyDi QA in the monolingual retrieval setting, but combining the sparse score and DPR score in sparse-dense hybrids perform even better. At the same time, sparse retrieval rely on lexical token matches and cannot do cross-lingual retrieval effectively without translating the query to the same language as the passage. To remove the need to use a machine translation system for simplicity, we rely on multilingual dense passage retrieval for cross-lingual retrieval.

For dense retrieval, we use FAISS (Johnson et al., 2019) with `IndexFlatIP`. For sparse retrieval, we use Pyserini (Yang et al., 2017; Lin et al., 2021) with BM25 with default parameters. We build separate indices for each language for both the dense and sparse setting. For each query, where we want to return K passage, we search for the top K passages globally in the dense indices in all languages, and search for the top K passages in the sparse index in the same language as the question.

We combine results from dense retrieval and sparse retrieval using the following algorithm, which we call Sparse-Corroborate-Dense. Our final ranked list consists of three ordered slices. The first slice consists of passages that are present in both dense and sparse retrieved lists, ranked in the same order as they appear in dense retrieval. The second slice consists of passages that are only in the dense ranked list and not in the sparse ranked list. The last slice consists of top passages in the sparse ranked list. The number of passages from the sparse hits that are allowed to influence the final ranked list is no more than $\lfloor max\_frac * K \rfloor$. We find this works better than the score normalization and combining approach in Mr. TyDi (Zhang et al., 2021) for cross-lingual retrieval. Figure 1 has an illustration of this algorithm running with $K = 5$ and $max\_frac = 0.6$ for a Bengali (bn) question. Please refer to Appendix A for code of the algorithm.

3.3 Reader

Instead of concatenating the question and all the passages in the input to the encoder like in the baseline, which we will call Fusion-in-Encoder, we use the Fusion-in-Decoder (FiD) approach (Izacard and Grave, 2020). In Fusion-in-Decoder, the encoder processes each of the *ctxs* passages independently adding special tokens *question: lang: title:* and *context:* before the question, title and text of each passage, while the decoder performs attention over the concatenation of the resulting representations of all the retrieved passages.

Independent processing of passages in the encoder allows to scale linearly to large number of contexts, while processing passages jointly in the decoder helps better aggregate evidence from multiple passages.

In order to semantically ground the entities across different languages together, we use Wikipedia language links to augment the data from retriever while training FiD based reader, like the CORA baseline. First, for each question in the MIA training set that comes from Natural Questions, we use the answer to search for the corresponding Wikipedia page using the Wikipedia API. Generally, only answers that are entities will have a result. This returns the titles of the Wikipedia articles in different languages, which we use as the answer in different languages. We use the DPR checkpoint trained with adversarial examples to retrieve English passages from the index.[4] For each English question-answer pair, we find corresponding entries in other

[4] https://github.com/facebookresearch/DPR#new-march-2021-retrieval-model

languages being evaluated in the task and generate $[Query_{eng}, Lang_{target}, Answer_{target}, Passages]$ tuples for training FiD. This data is augmented to the original training data provided by the retriever.

4 Results

For training the dual encoder, we use the official training data without any hard negatives with the same hyperparameters as the baseline dual encoder (Asai et al., 2021). For training Fusion-in-Decoder, we combine the all of the retrieval results with sampled Wikipedia language link augmented passages such that the total percentage of training examples from either source is 50%. We use the baseline retrieval results instead of mLUKE-retrieved results to develop the retriever and reader in parallel. We use learning rate of 0.00005 with linear learning schedule with a weight decay of 0.01 using the AdamW optimizer. The context size (number of passages) in the final submission is 20 passages. Note for retrieval we use $K = 60$ to use the same retrieval results for different context size experiments, but in the final submitted system take the top 20 from this list for the reader. We use $max\_frac = 0.2$ for Sparse-Corroborate-Dense. We use the best checkpoint on the development set for both components.

4.1 Main Results

We first report end-to-end results using our best system compared to the baseline in Table 2 for the development set and Table 3 for the test set. On the development set, we obtain macro-averaged F1 score of 43.46 across all languages on XOR-TyDi QA, an improvement of 3.70 F1 points over 39.76 obtained by baseline 1. We obtain macro-averaged F1 score of 21.99 across all languages on MKQA, an improvement of 4.61 F1 points over 17.38 obtained by baseline 1. On the test set, we observe fairly consistent results compared to the development set. On XOR-TyDi QA, our system and baseline 1 obtains 40.93 and 37.95 respectively, an improvement of 2.98 F1 points. On MKQA, our system and baseline 1 obtains 22.29 and 17.14 respectively, an improvement of 5.15 F1 points. On both the development set and test set, we outperform the baseline on all languages except for Khmer (`km`) on MKQA.

We observe our system frequently retrieves irrelevant passages for Khmer through qualitatively sampling some passages retrieved for Khmer questions, providing little chance for the reader to find the answer. mLUKE uses 24 languages for pretraining and does not include Khmer, making it difficult to align entities in Khmer. Furthermore, even if we use the baseline retrieval results, we still see a large drop in reader effectiveness when we switch to Fusion-in-Decoder from row (iii) to (ii) in Table 4. We only have 3101 rows in the training data for Khmer for our reader all from Wikipedia language links, out of 275990 rows in total.

On the surprising languages Tagalog (`tl`) and Tamil (`ta`), we outperform the baseline by a large margin. Perhaps surprisingly, this large improvement cannot be attributed to the presence of Tagalog and Tamil passages in our corpus, since in our best submission, for example, out of the 350 Tamil questions, only one question has a retrieved passage in Tamil in the top results that are fed to the reader. Instead, the system is able to generate correct answers from English passages.

4.2 Analysis

Ablation Studies We conduct ablation studies on our system in Table 4. We find the biggest gain comes from switching Fusion-in-Encoder (FiE) in the baseline to Fusion-in-Decoder (FiD) from row (iii) to row (ii), even though we did not increase the number of passages for Fusion-in-Decoder and kept it at 20 for the final system. The second largest gain comes from switching mBERT to mLUKE from row (ii) to row (i). Finally, the smaller gain comes from switching dense retrieval only to Sparse-Corroborate-Dense, from row (i) to mLUKE + SP + FiD. We study each of the components in greater detail below.

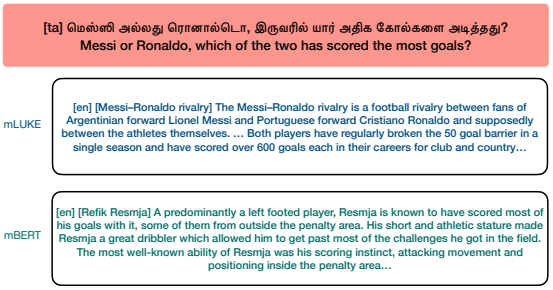

Figure 2: The top passage for a Tamil question retrieved by mBERT and mLUKE. We see mLUKE is able to find English passages related to entities Messi and Ronaldo, but mBERT struggles and only finds a general passage related to another unrelated soccer player related to goal scoring.

	XOR-TyDi QA F1								MKQA F1												
System	ar	bn	fi	ja	ko	ru	te	Avg	ar	en	es	fi	ko	ms	ja	km	ru	sv	tr	zh_cn	Avg
Baseline 1	51.29	28.72	44.35	43.21	29.84	40.68	40.19	39.76	8.77	27.86	24.92	23.25	8.28	22.64	15.18	**5.73**	14.00	24.13	20.60	13.14	17.38
Our submission	**54.84**	**30.68**	**47.41**	**47.29**	**33.90**	**43.13**	**47.00**	**43.46**	**13.34**	**39.57**	**29.74**	**24.73**	**12.14**	**27.44**	**18.97**	2.57	**19.36**	**28.26**	**25.52**	**22.29**	**21.99**

Table 2: End-to-end development set results. Baseline 1 and our submission obtain overall macro-averaged F1 scores of 28.57 and 32.73 respectively. Our submission outperforms the baseline on all languages except Khmer (km) on MKQA.

	XOR-TyDi QA F1								MKQA F1													Sup	
System	ar	bn	fi	ja	ko	ru	te	Avg	ar	en	es	fi	ko	ms	ja	km	ru	sv	tr	zh_cn	Avg	ta	tl
Baseline 1	49.66	33.99	39.54	39.72	25.59	40.98	36.16	37.95	9.52	36.34	27.23	22.70	7.68	25.11	15.89	**6.00**	14.60	26.69	21.66	13.78	17.14	0.00	12.78
Our submission	**55.33**	**30.48**	**41.01**	**43.45**	**31.21**	**42.62**	**42.40**	**40.93**	**12.67**	**39.63**	**30.85**	**25.22**	**12.18**	**29.09**	**20.49**	2.36	**18.82**	**29.62**	**26.16**	**22.60**	**22.29**	**20.75**	**20.95**

Table 3: End-to-end test set results. Baseline 1 and our submission obtain overall macro-averaged F1 scores of 27.55 and 31.61 respectively. "Sup" indicates the surprise languages. Our submission outperforms the baseline on all languages except Khmer (km) on MKQA.

Entity Representations To evaluate the passage retrieval component for XOR-TyDi QA, we measure MRR@60 and Recall@60. We picked 60 because it is the near the maximum number of passages we can feed into Fusion-in-Decoder bound by the GPU memory. For each question, to determine if a passage is relevant, we use a heuristic. First, we find the universe set of answers for the questions, which not only contain answers in the same language, but also possibly answers in English using the English answer in the XOR-English Span task (Asai et al., 2020). We check if the normalized answer is a substring of the passage text, and if so, we mark the passage as relevant. Note that this is a proxy for measuring passage relevance, since answers may not necessarily be exact spans / substrings or the same answer may appear as a substring in a non-relevant passage, but we found it to correlate well with end-to-end effectiveness. We see from Table 5 that overall using mLUKE improves passage retrieval effectiveness. Qualitatively, we also find examples where the dual encoder trained with mLUKE can find passages cross-lingually with the relevant entity whereas that trained with mBERT could not. In Figure 2, we see mLUKE can retrieve an English top passage about the soccer players Messi and Ronaldo asked in Tamil, but mBERT returns just an English passage about another soccer player not relevant to the question.

Dense-Sparse Hybrids Next, we evaluate the benefits of using dense retrieval in conjunction with sparse retrieval as opposed to using only dense retrieval in Table 6. The dense retriever here is mLUKE. We see that dense retrieval always works better than sparse retrieval when used independently, and the score combination approach used

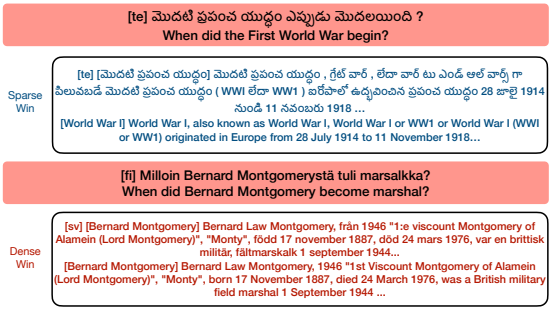

Figure 3: Here we see a highly relevant passage found by sparse monolingual retrieval that is not found by dense retrieval, and a relevant passage found by dense retrieval cross-lingually that is not found by sparse retrieval.

in Mr. TyDi (Zhang et al., 2021) does not outperform dense retrieval in recall, but does improve the MRR. We use Sparse-Corroborate-Dense, which piggybacks on dense retrieval results, but boosts the ranking of some passages in dense retrieval, and add in additional passages not found by dense retrieval to the end of the top-K list. Compared to dense only, it is better on both MRR and recall. When both dense and sparse retrieval finds the same passage, it is a strong signal the passage is relevant. Nonetheless, sparse retrieval can still find passages that dense retrieval cannot find, and adding these to the candidate passage list passed to the reader can provide additional relevant evidence passages. In Figure 3, we see sparse retrieval can find a highly relevant passage related to World War I in Telugu (te) to the Telugu question that cannot be found by dense retrieval, and dense retrieval can find a passage related to Bernard Montgomery cross-lingually in Swedish (sv) to a Finnish (fi) question that cannot be found by sparse retrieval – they can complement each other.

System	XOR-TyDi QA F1								MKQA F1											
	ar	bn	fi	ja	ko	ru	te	Avg	ar	en	es	fi	ko	ms	ja	km	ru	sv	tr	zh_cn	Avg
mLUKE + SP + FiD	54.84	**30.68**	**47.41**	**47.29**	**33.90**	**43.13**	**47.00**	**43.46**	**13.34**	**39.57**	**29.74**	24.73	**12.14**	**27.44**	**18.97**	**2.57**	**19.36**	**28.26**	**25.52**	**22.29**	**21.99**
(i) mLUKE + FiD	**54.93**	29.56	46.88	45.76	33.16	42.03	46.28	42.66	13.23	38.01	29.57	**25.36**	11.45	27.28	18.37	2.53	18.59	28.22	25.43	21.98	21.67
(ii) mBERT + FiD	53.19	29.25	46.97	43.25	30.38	42.79	44.22	41.44	10.94	37.42	28.18	21.89	9.63	27.20	15.00	2.11	16.41	26.96	21.86	20.24	19.82
(iii) mBERT + FiE	49.71	29.15	42.72	41.20	30.64	40.16	38.57	38.88	8.95	33.87	25.08	21.15	6.72	24.55	15.27	6.05	15.60	25.53	20.44	13.71	18.07

Table 4: Ablation studies on the development sets. mLUKE + SP + FiD is our submission with mLUKE + Sparse-Corroborate-Dense. (i) mLUKE + FiD only relies on dense retrieval, and we observe a slight decrease in the F1 score of most languages compared with our submission. (ii) mBERT + FiD changes the retriever to mBERT, and we observe a larger drop in F1 score compared to mLUKE in row (i). (iii) mBERT + FiE changes Fusion-in-Decoder to Fusion-in-Encoder as in the baseline and we see an even larger drop in F1 score compared with row (ii).

Model	MRR@60							
	ar	bn	fi	ja	ko	ru	te	Avg
mBERT (Devlin et al., 2019)	0.106	0.026	0.069	0.031	0.023	0.057	**0.050**	0.362
mLUKE (Ri et al., 2021)	0.106	**0.028**	**0.076**	**0.035**	**0.027**	**0.122**	0.042	**0.372**
Model	Recall@60							
	ar	bn	fi	ja	ko	ru	te	Avg
mBERT (Devlin et al., 2019)	0.185	0.057	0.130	0.073	0.050	**0.118**	0.075	0.689
mLUKE (Ri et al., 2021)	**0.189**	**0.065**	**0.133**	**0.078**	**0.056**	0.056	**0.079**	**0.723**

Table 5: MRR@60 and Recall@60 of passage retrieval for XOR-TyDi QA dev set for different pretrained language models.

Methodology	MRR@60							
	ar	bn	fi	ja	ko	ru	te	Avg
Sparse Only	0.088	0.023	0.640	0.024	0.018	0.051	0.032	0.299
Dense Only	0.106	0.028	0.076	0.035	0.027	0.058	0.042	0.372
Combine Score (Zhang et al., 2021)	0.113	0.029	0.076	0.032	0.023	0.060	0.048	**0.382**
Sparse-Corroborate-Dense	0.110	0.029	0.074	0.032	0.026	0.063	0.049	**0.382**
Methodology	Recall@60							
	ar	bn	fi	ja	ko	ru	te	Avg
Sparse Only	0.172	0.045	0.120	0.063	0.044	0.098	0.070	0.611
Dense Only	0.189	0.065	0.133	0.078	0.056	0.122	0.079	0.723
Combine Score (Zhang et al., 2021)	0.178	0.059	0.118	0.070	0.046	0.107	0.076	0.652
Sparse-Corroborate-Dense	0.192	0.065	0.136	0.078	0.057	0.124	0.080	**0.733**

Table 6: Comparison of various dense-sparse hybrid strategies for original XOR-TyDi QA dev set. The dense retrieval dual encoder used is mLUKE. max_frac used for Sparse-Corroborate-Dense is 0.2.

Fusion-in-Decoder We want to understand the effect of increasing the number of passages sent to the reader by comparing the effectiveness of the reader when there are 20 passages versus 60 passages. Intuitively, there could be relevant passages found in positions 21-60, which should strengthen the evidence needed to output the final answer. From Table 7 we observe using more evidence passages consistently improve results, and this scaling advantage is key over Fusion-in-Encoder. However, due to time limitations, we only used the 20 passages setting for the final shared task submission.

5 Conclusion

We describe our submission for the MIA 2022 Shared Task and detail some experiments we perform to improve specific components of the system. We find that using mLUKE (Ri et al., 2021), a pretrained language model that models entities, combing dense and sparse results using Sparse-Corroborate-Dense, and Fusion-in-Decoder, are effective for improving the effectiveness for cross-lingual question answering over the baseline.

Number of Passages	XOR-TyDi QA		MKQA	
	EM	F1	EM	F1
20	31.63	38.06	16.15	20.21
60	33.74	41.29	17.31	21.51

Table 7: Exact Match (EM) and F1 score for different number of passages on the development sets from mLUKE retrieved passages.

6 Acknowledgement

We thank Yinfei Yang, Wei Wang, and Jinhao Lei for their insightful discussions and feedback on early versions of the paper.

References

Akari Asai, Jungo Kasai, Jonathan H Clark, Kenton Lee, Eunsol Choi, and Hannaneh Hajishirzi. 2020. XOR QA: Cross-lingual open-retrieval question answering. *arXiv preprint arXiv:2010.11856*.

Akari Asai, Xinyan Yu, Jungo Kasai, and Hannaneh Hajishirzi. 2021. One question answering model for many languages with cross-lingual dense passage retrieval. In *NeurIPS*.

Jonathan H Clark, Jennimaria Palomaki, Vitaly Nikolaev, Eunsol Choi, Dan Garrette, Michael Collins, and Tom Kwiatkowski. 2020. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

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.

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proc. of EMNLP*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural Questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: An easy-to-use python toolkit to support replicable ir research with sparse and dense representations. *arXiv preprint arXiv:2102.10073*.

Shayne Longpre, Yi Lu, and Joachim Daiber. 2020. MKQA: A linguistically diverse benchmark for multilingual open domain question answering. *arXiv preprint arXiv:2007.15207*.

Ryokan Ri, Ikuya Yamada, and Yoshimasa Tsuruoka. 2021. mLUKE: The power of entity representations in multilingual pretrained language models. *arXiv preprint arXiv:2110.08151*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mT5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

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)*. Association for Computational Linguistics.

Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the use of lucene for information retrieval research. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pages 1253–1256.

Xinyu Zhang, Xueguang Ma, Peng Shi, and Jimmy Lin. 2021. Mr. TyDi: A multi-lingual benchmark for dense retrieval. *arXiv:2108.08787*.

A Sparse-Corroborate-Dense Algorithm

Here is the precise algorithm for Sparse-Corroborate-Dense.

```
def sparse_corroborate_dense(
dense_hits: List[Tuple[str, float]],
sparse_hits: List[Tuple[str, float]],
max_frac: float, K: int):
  dense_docid_to_idx = {
    tup[0]: idx for idx, tup in
  enumerate(dense_hits)
  }

  RESERVED_SPARSE_SLOTS = min(
    int(max_frac * K),
    len(sparse_hits)
  )
  final_hits = []
  docids_added = set()
  backfill_sparse_hits = []

  # Go through top sparse results, if
  sparse hit is also in dense, push it
   to front, else, put it in backfill
  for docid, sparse_score in
  sparse_hits:
    if docid in dense_docid_to_idx:
      final_hits.append(
        dense_hits[dense_docid_to_idx[
  docid]]
      )
      docids_added.add(docid)
      RESERVED_SPARSE_SLOTS -= 1
    else:
      backfill_sparse_hits.append([
  docid, sparse_score])

  # Add rest of dense ids
  i = 0
  while len(final_hits) < K -
  RESERVED_SPARSE_SLOTS and i < len(
  dense_hits):
    if dense_hits[i][0] not in
  docids_added:
      final_hits.append(dense_hits[i])
      docids_added.add(
        dense_hits[i][0]
      )
      i += 1

  final_hits.extend(
    backfill_sparse_hits[:K - len(
  final_hits)]
  )

  return final_hits
```