Natural Language Processing Project - Group 47

Dominique Julien Rustler **Armin Irvije** **Simon August Mørk**
{mkp570, rjc588, jvl527}@alumni.ku.dk

Table 1: Data statistics for the training and validation sets. "Avg. Length" and "Vocab Size" (measured in words) are shown as (Question / Context).

Split	Language	Samples	Avg. Length (Qst / Ctx)	Vocab Size (Qst / Ctx)
Train	Arabic	2,558	6.33 / 103.40	5,399 / 27,194
	Korean	2,422	5.73 / 96.56	6,114 / 25,052
	Telugu	1,355	5.66 / 87.58	2,411 / 16,238
Val	Arabic	415	6.31 / 103.22	1,180 / 9,087
	Korean	356	5.64 / 95.36	1,073 / 7,598
	Telugu	384	5.99 / 106.27	741 / 7,901

1 Week 36 - Rule Based Classifier

We analyze the `tydi-xor-rc` (Asai et al., 2021; Clark et al., 2020; Muller et al., 2023) dataset for Arabic (Ar), Korean (Ko), and Telugu (Te). Table 1 summarizes training and validation statistics computed with a language-specific word-level tokenizer (details in Appendix A.1). Shown below are the five most common words, together with their translations and frequencies:

- **Arabic:** في (in, 593), من (from, 586), متى (when, 535), ما (what, 442), هو (he, 349).
- **Korean:** 무엇 (what, 607), 가장 (most, 529), 언제 (when, 433), 어디 (where, 316), 누구 (who, 311).
- **Telugu:** ఎవరు (who, 274), ఏది (which is, 192), ఎన్ని (how many, 165), ఎప్పుడు (when, 154), ఏ (A, 142).

where translations were generated using `Google Translate`[1]. These common words include interrogatives (e.g., who, what, when) and common function words (e.g., in, from, he), which is typical for question data.

1.1 Rule-Based Classifier

We built a Rule-Based Classifier (RBC) that operates on a English-translated question (via NLLB model) and the context, and labels them as answerable if (1) its word overlap with the context exceeding 30% (after stopword and punctuation removal) or (2) a named-entity or noun overlap is detected. Details in Appendix A.3. We evaluated performance using Accuracy and F1 (Section 1.1), as F1 accounts for class imbalance (Ko. 94.7 %, Te. 75.8 %, Ar. 87.5 % answerable). The RBC achieves strong F1 scores but is generally outperformed by a naive "always answerable" baseline (see Section 1.1). For Telugu, however, it performs better, suggesting that incorporating semantic cues helps identify unanswerable cases in the most imbalanced language subset.

Language	Rule-Based		Naive Baseline	
	Acc.	F1	Acc.	F1
Arabic	0.7880	0.8791	0.8747	0.9332
Korean	0.8006	0.8878	0.9466	0.9726
Telugu	0.7917	0.8726	0.7578	0.8622

Table 2: Validation Accuracy (↑) and F1-Score (↑) of the Rule-Based Classifier and Naive Baseline.

2 Week 37 - Language Models

We implemented three language models of increasing complexity for English contexts and Arabic, Korean, and Telugu questions: (1) a Unigram model as a frequency baseline, (2) a Bigram model capturing local dependencies, and (3) an LSTM model for long-range context modeling. Our n-gram models use Laplace smoothing to handle unseen n-grams. We applied the tokenizer from multilingual BERT as it segments words into meaningful subwords, reducing out-of-vocabulary problems while keeping a shared multilingual vocabulary. To limit sparsity, only tokens occurring in the training data were retained. Details in Appendix B.1.

2.1 Evaluation and Discussion

Models were evaluated using Perplexity (PPL), which measures how well a model predicts the next token given its context (preceding tokens). Results are shown in Table 3. The LSTM outperforms

[1] https://pypi.org/project/deep-translator/

Table 3: Perplexity Scores (↓) for Language Models on the validation set.

Model	English	Arabic	Korean	Telugu
Unigram	1630.19	409.06	303.26	205.72
Bigram	2597.64	612.92	118.81	86.42
LSTM	**165.33**	**141.86**	**30.19**	**33.99**

n-gram models across all languages by learning representations that captures sequential dependencies. The improvement is most pronounced for English contexts (165.33 vs. 1630.19), where LSTMs leverage longer sequences (avg. 103 words) more effectively. The lower perplexity (30-142) of the questions might be due to their brevity (avg. 6 words) and predictable structure, benefiting both n-gram and neural models. Counterintuitively, Bigrams perform worse than unigrams for English and Arabic due to data sparsity and the limitations of Laplace smoothing. The much larger space of possible bigrams leads to many unseen pairs, resulting in excessive smoothing and thus degraded estimates for observed pairs. The smaller vocabularies in Korean and Telugu mitigate this effect, allowing Bigrams to outperform unigrams as expected. Alternative smoothing methods (e.g., Kneser-Ney) could address this issue. See Appendix B.2 for results with alternative smoothing, reduced vocabulary and tokenization methods.

3 Week 38 - ML Classifiers

We implemented three hierarchically more complex classifiers to predict whether a question is answerable given its context, for questions in Arabic, Korean and Telugu. First, a Logistic Regression baseline as a non-neural baseline that uses TF-IDF features to measure the relative frequency and importance of word and n-gram overlap of the question and context. Next, a Bidirectional LSTM (BiLSTM) is used, as it combines a bidirectional architecture for contextual understanding with LSTM cells for long-range dependency modeling. Finally, we fine-tuned a Multilingual BERT (mBERT)[2] model, leveraging its advanced transformer architecture and context understanding acquired from multilingual pre-training. All models used weighted Cross-Entropy Loss to address class imbalance to penalize misclassifications of the minority class. Details in Appendix C.1.

[2] `bert-base-multilingual-cased`

Table 4: Performance of the classifiers on the validation set using Accuracy (↑) and F1-Score (↑).

Model	Arabic		Korean		Telugu	
	Acc.	F1	Acc.	F1	Acc.	F1
LogReg	0.9518	0.9731	0.9466	0.9726	0.7630	0.8594
BiLSTM	**0.9807**	**0.9889**	0.9494	0.9738	0.8281	0.8862
mBERT	**0.9807**	**0.9889**	**0.9775**	**0.9881**	**0.9115**	**0.9439**

3.1 Evaluation and Discussion

Similar to Week 36, we report Accuracy and F1-Scores. Results are shown in Table 4. We observe a clear trend: Neural models (BiLSTM, mBERT) outperform the LogReg baseline, showing that semantic and sequential features are superior to lexical overlap for answerability. Furthermore, mBERT outperforms BiLSTM, showing the strength of deep contextualized representations. Interestingly performing identically on Arabic. All models perform as well as or better than a naive "always answerable" baseline. Arabic achieved the best overall F1-Score, while Telugu was the worst. Crucially, however, Telugu showed the largest relative improvement over the naive baseline (≈9.5%) compared to Arabic (≈6%) and Korean (≈1.6%). This substantial gain likely stems from a higher proportion of diverse unanswerable samples in the Telugu validation set, which better showcases the model's learned ability to distinguish subtle differences between answerable and unanswerable questions.

4 Week 39 - Open QA

We implemented three Open QA settings for Telugu using the `google/mt5-small`[3] model and tokenizer from Hugging Face to analyze the impact of context, and answer language: (1) with context ($Q{+}C{\rightarrow}A_{Te}$), (2) without context ($Q{\rightarrow}A_{Te}$), and (3) cross-lingual with English answers ($Q{\rightarrow}A_{En}$). Setting 1 and 2 used a smaller subset of Telugu data, which was limited and mostly contained unanswerable samples, while Setting 3 used the full dataset with English answers, with mostly answerable questions. Each setting used a detailed prompt that explained the setting to leverage the model's pre-trained multilingual contextual understanding. Detailed prompts and implementation in Appendix D.1.

4.1 Results and Discussion

We evaluate models using ROUGE-1 for unigram (token-level) overlap and ChrF++ for character-

[3] https://huggingface.co/google/mt5-small

Table 5: Open QA Generation Performance on ROUGE-1 (↑) and ChrF++ (↑) on the validation set for Answerable (Ans.), Unanswerable (Unans.) and Overall data.

Metric Condition	Setting 1 (Q+C → A_{Te})	Setting 2 (Q → A_{Te})	Setting 3 (Q → A_{En})
ROUGE-1 (Ans.)	0.000	0.143	0.137
ROUGE-1 (Unans.)	0.075	0.183	0.828
ROUGE-1 (Overall)	0.070	0.180	0.304
ChrF++ (Ans.)	4.70	28.16	14.10
ChrF++ (Unans.)	41.60	85.84	87.64
ChrF++ (Overall)	40.78	84.67	31.05

level matching, which is well suited to morphologically rich languages like Telugu. Table 5 shows the results. All settings achieve better metric scores for unanswerable than for answerable questions. This gap likely reflects the nature of the questions (see Appendix D.2). Unanswerable questions often test general facts that are likely frequent in the model's pre-training, while answerable ones tend to require domain-specific information (likely less represented in pre-training) or may have multiple valid but context-dependent answers. Additionally, answerable training examples appear simpler, which may limit the model's ability to generalize to more complex questions. However, for Setting 1, where context is provided, this alone does not explain the poor performance. We attribute it to the predominance of unanswerable samples, which likely confuses the model and causes it to treat the context as noise, leading to hallucinated answers.

The models are also able to answer correctly without context by relying on knowledge from pretraining: Setting 1 achieved 29% exact match on contextually unanswerable items, while Settings 2 and 3 reached 79% and 26%, respectively (Example outputs in Appendix D.3). Setting 3 obtained the highest ROUGE-1, possibly benefiting from more training data[4], while Setting 2 achieved the highest ChrF++. This might reflect ROUGE-1's focus on token overlap and ChrF++'s sensitivity to character-level variation, which benefits Telugu. More notably, despite Settings 1 and 2 sharing the same task and data (but Setting 2 lacking context), they show a huge performance difference, further suggesting that the context in Setting 1 acts as a noise rather than as a useful signal. We also observed duplicated questions across splits (sometimes with different answers), which may affect evaluation quality.

[4]We realized, that comparison is limited due to different training and validation data, however we did not have the time to retrain the model using the same data.

Table 6: Span-based Metrics (Exact Match (EM) (↑), $F1_{span}$ (↑), and $F1_{token}$ (↑)) on the Validation Set

Language	Model	EM	$F1_{span}$	$F1_{token}$
Arabic	BiLSTM	0.1614	0.2030	0.5058
	mBERT	0.4916	0.4900	0.6994
	mDeBERTa	**0.5325**	**0.5027**	**0.7234**
Korean	BiLSTM	0.1489	0.2077	0.4945
	mBERT	0.4916	0.5165	0.7034
	mDeBERTa	**0.5225**	**0.5455**	**0.7485**
Telugu	BiLSTM	0.1901	0.1623	0.4721
	mBERT	0.4193	0.4267	0.6302
	mDeBERTa	**0.4219**	**0.4731**	**0.6636**

5 Week 40 - Span Based QA

In this week, we implemented three hierarchically more complex models for span-based QA in Arabic, Korean, and Telugu by using the BIO (Begin–Inside–Outside) tagging scheme to represent answer spans. We masked non-context tokens (label = –100) to exclude them from loss computation. Unanswerable questions are represented by sequences of 'O' tags. To establish a clear performance hierarchy for analysis, we trained three multilingual sequence models: a BiLSTM with a `MultiheadAttention` layer as a compact baseline, mBERT as a pretrained Transformer benchmark, and mDeBERTa (`microsoft/mdeberta-v3-base`) as a higher-capacity variant. All models were trained on combined multilingual data to exploit cross-lingual patterns and increase sample diversity. A weighted Cross-Entropy Loss was used to tackle the extreme class imbalance (predominantly "O" tokens). Details in Appendix E.1.

5.1 Results and Discussion

We evaluated our models using Exact Match (EM), that requires a perfect match of the entire predicted BIO sequence with the true sequence. $F1_{span}$ measures span-level performance, computing the F1-Score based on the overlap between predicted and gold answer spans. We also report $F1_{token}$ (macro F1 over B, I, and O classes) to capture overall labeling quality, as accuracy would be uninformative due to the dominance of 'O' tokens. Results are provided in Table 6.

The results show that transformer models outperform the simpler BiLSTM across all metrics, with mDeBERTa performing best. EM was lowest for transformers in Telugu, likely because of limited data and many unanswerable samples made learning exact spans difficult, while for BiLSTM, EM for Telugu was the highest which might re-

Lang.	RBC		mBERT (Avg.)	
	Acc.	F_1	Acc.	F_1
English	1.0000	1.0000	0.8	0.8816
Danish	0.7000	0.8235	0.8667	0.9279
German	0.5000	0.6667	0.8	0.8845

Table 7: Test set Accuracy (↑) and F1-Score (↑) of RBC and averaged mBERT.

flect a bias toward predicting unanswerable samples. $F1_{span}$ was highest for Korean and lowest for Telugu, indicating that span extraction becomes harder with less data. The gap between $F1_{token}$ and $F1_{span}$ shows that even with strong token labeling, exact span prediction remains challenging.

6 Week 41+ - Evaluation on Test set

For the final task, we created a test set (30 questions; 10 each in English, Danish, and German) to test the generalization capabilities of our created models on new data and languages, it was not trained on. However, due to the small sample size, evaluation provides rather an indicative measure of performance. The test set consists of mostly answerable questions (9/10 for English and Danish; 8/10 for German). Questions include general factual questions, but also more complex questions that cannot easily be answered without context by using knowledge from pre-training.

6.1 Answerability Classifier (Week 36 & 38)

We evaluated our Rule-Based Classifier (RBC, Section 1) and the fine-tuned multilingual BERT (mBERT, Section 2), which was our best-performing answerability model. For mBERT, we report results averaged across its language-specific variants (Arabic, Korean, Telugu). For the RBC, each question was first translated to English. As shown in Section 6.1, the RBC achieves perfect scores on English but performs poorly on Danish and German, reflecting the limitations of rule-based heuristics. The mBERT model, performs more consistently across languages and clearly surpasses the RBC on Danish and German, showing the advantage of learned representations.

6.2 Open QA (Week 39)

We chose the mt5 model from Setting 3, which predicts the English answer given our test set questions, as it was trained on the largest dataset and thus expected to generalize best. Table 8 shows ROUGE-1 and ChrF++ scores near zero for all languages, indicating failed cross-lingual transfer. Examination of generated outputs reveals the model produced completely unrelated answers across all languages (e.g., “Alaska” for “tallest mountain in Iran”, “Dynasty” for “Queen Margrethe’s full name”), suggesting it generated arbitrary facts from its pre-training rather than attempting to answer the question.

Language	Rouge-1	ChrF++
English	0.000	3.426
Danish	0.025	3.1759
German	0.0	2.4509

Table 8: Open QA Generation Performance using Rouge-1 and ChrF++ on the test set

Language	EM	$F1_{span}$	$F1_{token}$
English	0.3	0.2105	0.8161
Danish	0.2	0.4444	0.7262
German	0.2	0.2222	0.7686

Table 9: Span-based Metrics (Exact Match (EM), $F1_{span}$, and $F1_{token}$) on the Test Set

6.3 Span-Based QA (Week 40)

For this week, we chose our mDeBERTa model as it was the overall best performing model for Week 40. Table 9 presents the results, which shows a strong contrast. Although $F1_{token}$ scores are high, indicating that the model can identify relevant tokens across all classes, the low EM and $F1_{span}$ scores show that precisely locating answer span boundaries remains a major challenge, despite the learned multilingual representations during pre-training.

6.4 General Observation

All models showed reduced performance on unseen zero-shot languages, indicating that while multilingual pre-training enables cross-lingual transfer, fine-tuning on typologically diverse target data remains crucial for optimal downstream results. The models’ ability to retain moderate performance nonetheless reflects the strength of their learned multilingual representations.

7 Conclusion

Transformer models (mDeBERTa, mBERT) consistently outperformed simpler baselines, with performance correlating to model and training data size. Results on new question languages (English, Danish, and German) revealed notable degradation, confirming that pre-training alone is insufficient without language and task specific fine-tuning.

Contribution of each group member

Generally, for Week 37-40, we decided to split up each of the k models to each of the k group members, with each member doing their own design choices for the models. For Week41, as mkp570 trained the models that were used to evaluate the test set, this part was done by him. For Week39, the task was very complex and each group member tried to achieve good results for their assigned setting. Every member created models that performed on the task, over a span of multiple weeks as the task was pretty difficult, thus every group member put lots of effort in achieving qualitative results for their assigned task. In the end the best model results for each task were achieved by mkp570, thus using this models for the final report. Before reporting the scores for the final report, mkp570 polished and finalized all models used and was rerunning them. The final report, was based on our previously weekly assignment, where each member actively performed. mkp570, then polished and aligned Week36 - Week38 and Week40 and Week41 to the new data. As Week 39 had no inital results, they were also created during this process. Altogether, each group member actively participated in meetings, discussions and put in effort into each week and the final report.

References

Akari Asai, Jungo Kasai, Jonathan Clark, Kenton Lee, Eunsol Choi, and Hannaneh Hajishirzi. 2021. XOR QA: Cross-lingual open-retrieval question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 547–564, Online. Association for Computational Linguistics.

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

Benjamin Muller, John Wieting, Jonathan Clark, Tom Kwiatkowski, Sebastian Ruder, Livio Soares, Roee Aharoni, Jonathan Herzig, and Xinyi Wang. 2023. Evaluating and modeling attribution for cross-lingual question answering. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 144–157, Singapore. Association for Computational Linguistics.

A Appendix - Week 36

A.1 Our Word-level Tokenizer

We implemented a custom `WordTokenizer` class to handle the distinct linguistic properties of the languages in our dataset. For Arabic, it utilizes `pyarabic.araby.tokenize` [5] to correctly segment morphologically rich words, and for Korean, it employs `konlpy.tag.Okt` [6] to identify meaningful phrases. Telugu tokenization is handled by `indicnlp.tokenize.trivial_tokenize` [7], while the English contexts are tokenized using NLTK's standard `word_tokenize` function. This approach provides more accurate word-level segmentation than simple whitespace splitting, which is crucial for these morphologically complex languages

A.2 Translation Google Translate vs. NLLB

Table 10: Analysis of most common words in questions.

Language	Word	Count	Google Translation	NLLB Translation
Arabic	في	593	in	In the
	من	586	from	I am from
	متى	535	when	When?
	ما	442	what	We are not.
	هو	349	he	He is the
Korean	무엇	607	what	What?
	가장	529	most	The most
	언제	433	when	When?
	어디	316	where	Where are you going?
	누구	311	who	Who?
Telugu	ఎవరు	274	who	Who is who?
	ఏది	192	which is	Whatever it is
	ఎన్ని	165	how many	How many?
	ఎప్పుడు	154	when	When?
	ఏ	142	A	No, not at all.

A.3 Rule-Based Classifier

Our Rule-Based Classifier operated on the translated questions in English, generated from the `facebook/nllb-200-distilled-600M`[8] model. Our rule based classifier first took the translated question and context and removed punctuation, provided by the `string.punctuation` function. We then employed two rules: (1) removing stopwords provided by `nltk` and lowercasing the question and context, to then calculate the intersection of the unique vocabulary in the question and context and (2) using `spacy` (via the

[5] https://pypi.org/project/PyArabic/
[6] https://konlpy.org/en/latest/
[7] https://github.com/anoopkunchukuttan/indic_nlp_library
[8] https://huggingface.co/facebook/nllb-200-distilled-600M

en_core_web_sm model) to extract all named entities (e.g., PERSON, GPE, DATE) and nouns (NOUN, PROPN) from both the question and context and check if there is an overlap. A question is considered answerable, if either the overlap from (1) is above a 0.3 threshold or if there was an overlap in (2).

B Appendix - Week 37

B.1 Model Implementation

N-Gram Models We implemented a Unigram (n=1) and Bigram (n=2) language model that uses the subword tokenizer from the multilingual Bert model (bert-base-multilingual-cased). We restricted the vocabulary to only tokens present in each language's training set to reduce data sparsity. We added Laplace smoothing to handle unseen n-grams in the validation set. The probability of an n-gram is calculated as:

$$P(w_i \mid w_{i-n+1}^{i-1}) \frac{C(w_{i-n+1}^{i}) + 1}{C(w_{i-n+1}^{i-1}) + V}$$

where C is a count function, V the vocabulary size and w_{i-n+1}^{i-1} denotes the sequence of $n-1$ context tokens $\langle w_{i-n+1}, \ldots, w_{i-1} \rangle$ immediately preceding the token w_i.

LSTM The LSTM also uses the bert-base-multilingual-cased tokenizer, with an embedding dimension of 300, 2 layers with an hidden size of 512, combining a total of 100,960,031 parameters. Our texts in the dataset are truncated after 128 tokens, which generally does not have any effect for Telugu, Arabic, and Korean, but for English. However, as English has in general more training data and for computational reasons, we have not increased nor implemented a sliding window approach. We trained the LSTM for 15 epochs on each language separately using language dependent learning rates (English: 0.001, Telugu: 0.1, Arabic/Korean: 0.01), the Adam optimizer and a batch size of 16.

B.2 Other Results

Table 11: Perplexity Scores (↓) for N-Gram models on the validation set using the top 1000 most common vocabularies.

Model Type	English	Arabic	Korean	Telugu
Unigram	57.97	224.40	214.77	205.72
Bigram	32.45	210.63	69.12	86.42

Fixed Vocabulary for n-gram Models Table 11 shows the results, limiting the vocabulary to the top 1000 most common tokens + the unknown token. We can observe that for Telugu nothing changed, as it was using a lower vocabulary anyway. For each language we can now observe the effect of the Bigram performing better than the Unigram model. However, limiting the vocabulary for a better perplexity score is not making the language model more capable.

Using Kneser-Ney Smoothing Bigram results of using Kneser-Ney smoothing for Arabic, Korean, and Telugu. English is omitted due to computational reasons. Note that Kneser-Ney cannot be computed for Unigram models as it is based on the continuation probability which calculates a word's likelihood based on the diversity of unique words that precede it, which is not given in a Unigram setting.

Table 12: Perplexity Scores (↓) for Bigram models on the validation set using Kneser-Ney smoothing.

Model Type	English	Arabic	Korean	Telugu
Bigram	–	185.16	33.48	31.25

Table 12 present the results. We can observe how close we get to the LSTM performance, even achieving better performance for Telugu, showing the power of this smoothing method.

Using Character level tokenization Our LSTM used the same architecture and hyperparameters as our subword level LSTM. Table 13 showing the

Table 13: Perplexity Scores (↓) for Language Models on the validation set using character level tokenization.

Model Type	English	Arabic	Korean	Telugu
Unigram	24.55	21.49	100.68	30.90
Bigram	12.70	13.60	34.97	12.27
LSTM	3.82	5.33	10.13	9.25

results. As the Vocabulary massively decreases (For n-grams: Telugu: 95, Arabic: 113, Korean: 828, English: 1254) we see massive reduction in perplexity and also the common pattern that Unigram performs worse than Bigram and worse than a trained LSTM. However, as we no longer use subwords, it is an entirely different task than before and perplexity scores cannot directly be compared.

C Appendix - Week 38

C.1 Implementation details

LogReg The Logistic Regression model was implemented using scikit-learn's LogisticRegression classifier with a TF-IDF vectorization pipeline. The input consisted of concatenated question and context text pairs, which were then transformed into TF-IDF feature vectors. For the TF-IDF features, we filtered out terms that appear in more than 90% of documents and removed further english stop words.
A grid search with 3-fold cross-validation was performed to optimize the following hyperparameters:

- ngram_range: {(1,1), (1,2), (1,3), (2,2), (2,3)}
- max_features: {5,000, 10,000, 50,000, 100,000, 200,000}
- C: {0.1, 1.0, 10.0, 100.0, 1,000.0, 10,000.0}

The F1-score was used as the evaluation metric for selecting the best hyperparameter combination. Separate models were trained for each language (Korean, Telugu, and Arabic). Final Parameters per language were:

- Korean: ngram_range=(1,3), max_features=100,000, C=1.0
- Telugu: ngram_range=(1,2), max_features=50,000, C=0.1
- Arabic: ngram_range=(1,3), max_features=100,000, C=10.0

Additionally we used `balanced` class weights in the Cross Entropy loss to address class imbalance.

BiLSTM The BiLSTM model was implemented using PyTorch with a custom architecture designed for sequence classification. The model processes tokenized text through an embedding layer with 256 dimensions, followed by three bidirectional LSTM layers of dimension 256, and concludes with a dropout (0.3 rate) and a fully connected classification head.
ext sequences were tokenized using the BERT multilingual tokenizer (bert-base-multilingual-cased) with the following configuration: max_length = 512, padding = True, truncation = True. Questions and contexts were passed to the tokenizer, creating a separator token between them. Truncation was used to match BERT's maximum sequence length. For Korean and Telugu truncation rates were below 0.6%, while for Arabic below 2.5%.
We used a weighted Cross-Entropy loss using weights computed by the `compute_class_weight` function by scikit-learn using the `balanced` strategy. We further used an Adam optimizer and trained for 15 epochs. We performed a grid search for our learning rate over {0.01, 0.001, 0.0001, 0..00001}. The best learning rate was selected based on validation F1-score, and separate models were trained for each language using language-specific class weights.

mBERT The mBERT model was fine-tuned using the Hugging Face Transformers library, leveraging the pre-trained bert-base-multilingual-cased model with a sequence classification head. We used the same tokenization as our BiLSTM. We used again a Weighted Cross Entropy Loss, implemented in a custom `WeightedLossTrainer`. We used AdamW optimizer, using a learning rate of $5e-5$, a batch size of 32 and trained for 3 epochs. We trained the model separately for each language.

D Appendix - Week 39

D.1 Implementation and Training Details

For all three Settings, we used the `google/mt5-small` model from Hugging Face and its corresponding tokenizer with a max length of 512 samples for setting 1 input and 128 for setting 1 output and setting 2 and 3 input and output with truncation and dynamic padding via `DataCollatorForSeq2Seq`. No truncation were observed.

For Setting 1 and 2, we filtered the dataset to only samples that have an question in Telugu and an answer in Telugu (`answer_inlang` field not null) resulting in 50 training (5 answerable and 45 unanswerable) and 100 validation (7 answerable, 93 unanswerable) samples. For Setting 3, we filtered the dataset to only samples that have an question in Telugu and an answer (`answer` field not null), resulting in 1335 training (1310 answerable, 45 unanswerable) and 384 validation (291 answerable, 93 unanswerable) samples.

Setting 1 We gave our model the following prompt: "`Given a question in Telugu: question_text, with a context in english: context_text. Generate an answer in Telugu:`"

Further, we used a Cross-Entropy Loss with a AdamW optimizer and a weight decay of 0.01 on a learning rate of 5e-4 and enabled predict_with_generate. We used a batch size of 8 and for 80 epochs.

Setting 2 We gave our model the following prompt: "`Given a question in Telugu: question_text; Generate an answer in Telugu:`"
Further, we used a Cross-Entropy Loss with a AdamW optimizer and a weight decay of 0.01 on a learning rate of 5e-4 and enabled predict_with_generate. We used a batch size of 8 and for 80 epochs.

Setting 3 We gave our model the following prompt: "`Given a question in Telugu: question_text; Generate an answer in English:`"
Further, we used a Cross-Entropy Loss with a AdamW optimizer and a weight decay of 0.01 on a learning rate of 5e-4 and enabled predict_with_generate. We used a batch size of 8 and for 50 epochs due to a larger dataset.

D.2 Questions Answerable vs Unanswerable

Answerable Questions (translated):

- **Question:** When did World War I begin?
  - **Answer:** 1914 (also 28 July 1914 with same question in Dataset)
  - Question were answer without context has multiple options
- **Question:** According to the 2011 census, how many houses are in Gottiproli village?
  - **Answer:** 511
  - Domain-specific information about Telugu cinema
- **Question:** When did Pakistan become independent?
  - **Answer:** 1947
  - From training set, showing a lower complexity than the validation set.

Unanswerable Questions (translated):

- **Question:** When did the East India Company come to India?
  - **Answer:** 1608
  - Historical Fact
- **Question:** According to Telugu Panchang, which English month is the start of the New Year?
  - **Answer:** March or April
  - Cultural knowledge about Indian calendar systems, probably encountered in multilingual pre-training data
- **Question:** What is the population density of New York City?
  - **Answer:** 28,491
  - Factual geographic information commonly found in reference materials

D.3 Example Outputs

Answerable

- **Question:** విశ్వామిత్రుడు ఏ స్వర్గాన్ని నిర్మించాడు? (Which heaven did Vishwamitra build?)
  - **Ground Truth:** త్రిశంకు (Trishanku)
  - **Answer (Setting 1):** శ్రీకాకుళం (Srikakulam)
  - **Answer (Setting 2):** త్రిశంకు (Trishanku)
  - **Answer (Setting 3):** Trishanku
- **Question:** సింగిరెడ్డి నారాయణరెడ్డి జ్ఞానపీఠ పురస్కారం ను ఎప్పుడు అందుకున్నాడు ? (When did Singireddy Narayana Reddy receive the Jnanpith Award?)
  - **Ground Truth:** 1988
  - **Answer (Setting 1):** తెలంగాణ లోని మెదక్ జిల్లా ఆందోల్ (Andol, Medak district, Telangana)
  - **Answer (Setting 2):** 1988
  - **Answer (Setting 3):** 1988

Unanswerable

- **Question:** మలేరియా వ్యాధి కి మందు కనిపెట్టిన శాస్త్రవేత్త ఎవరు? (Which scientist discovered the cure for malaria?)
  - **Ground Truth:** హన్స్ ఆండర్సాగ్ (Hans Andersen)
  - **Answer (Setting 1):** ఫ్రాన్స్ (France)
  - **Answer (Setting 2):** జేపీ మోర్గాన్ ఛేజ్ టవర్ (JPMorgan Chase Tower)
  - **Answer (Setting 3):** Sir William Herschel

- **Question:** ఈస్ట్ ఇండియా కంపెనీ భారతదేశంలోకి ఎప్పుడు వచ్చింది? (When did the East India Company come to India?)
  - **Ground Truth:** 1608
  - **Answer (Setting 1):** 1608
  - **Answer (Setting 2):** 1608
  - **Answer (Setting 3):** 1608

E Appendix - Week 40

E.1 Implementation and Training

BiLSTM This model consisted of a trainable embedding layer (300-dimensional), a 2-layer bidirectional LSTM with hidden size 256, a 4-head multihead self-attention mechanism, and a linear classifier for token-level predictions. The model had approximately 50 million trainable parameters. It further used the mBERT tokenizer for tokenization with a maximum sequence of 512 tokens.

mBERT The `bert-base-multilingual-cased` model (loaded from Hugging Face) was fine-tuned with a token classification head. This model contains approximately 178 million parameters and has been pretrained on 104 languages using masked language modeling. We used the model corresponding tokenizer with a maximum sequence of 512 tokens.

mDeBERTa The `microsoft/mdeberta-v3-base` model (loaded from Hugging Face) was fine-tuned with a token classification head. This model contains approximately 278 million parameters and employs disentangled attention mechanisms with enhanced mask decoders, representing the highest-capacity architecture in our hierarchy. We used the model corresponding tokenizer with a maximum sequence of 512 tokens.

All models were trained for 15 epochs with early stopping (with patience 5). We used a batch size of 8 for training and 16 for evaluation. Further our transformers first performed 500 warmup steps. We also used a Weight decay of 0.01. For computational reasons, we used FP16 training. Our BiLSTM further used a training rate of 10^{-3} and class weights for the weighted cross entropy loss of $[0.5, 8.0, 4.0]$ for O/B/I tags. mBERT used a learning rate of $3 \cdot 10^{-5}$ and class weights of $[0.5, 8.0, 4.0]$, while mDeBERTa used a learning rate of $2 \cdot 10^{-5}$ and class weights of $[0.7, 3.0, 1.8]$. All models were trained on the combined dataset from Arabic, Korean and Telugu. The training set comprised 6,335 examples (5,972 answerable and 363 unanswerable, corresponding to 94.3% and 5.7% respectively), while the validation set contained 1,155 examples (991 answerable and 164 unanswerable, corresponding to 85.8% and 14.2% respectively). For each token in the context, BIO tags were assigned based on character-level overlap with the gold answer span. The procedure maintained an `is_in_answer` flag to ensure that the first overlapping token received a "B" label and subsequent overlapping tokens received "I" labels. Tokens outside the answer span were labeled as "O". For unanswerable questions, all context tokens were labeled "O".

F Hardware and Software

We used an Intel Ultra 7 CPU and a RTX 5080 GPU. Further, Python 3.13.7 with Torch 2.8.0+cu129 was used. We used `SEED = 42`.

UCPH's AI declaration

Declaration of using generative AI tools
☒ **I/we have used generative AI as an aid/tool** *(please tick)* ☐ **I/we have <u>NOT</u> used generative AI as an aid/tool** *(please tick)* *If generative AI is permitted in the exam, but you haven't used it in your exam paper, you just need to tick the box stating that you have not used GAI. You don't have to fill in the rest.*
List which GAI tools you have used and include the link to the platform (if possible): - *GitHub Copilot [https://github.com/features/copilot]* - *Claude [https://claude.ai/]* - *ChatGPT [https://chatgpt.com/]*
Describe how generative AI has been used in the exam paper: *1) Purpose (what did you use the tool for?)* *2) Work phase (when in the process did you use GAI?)* *3) What did you do with the output? (including any editing of or continued work on the output)* *We used GitHub Copilot and Claude for helping with coding. Claude was used for debugging and testing to help resolve errors in our code, while GitHub copilot was used during active coding by generating suggestions. The suggested code and fixes were reviewed and corrected and modified if needed. The debugging information by Claude were used to understand the errors and to fix them.* *ChatGPT was used during writing for the report to provide suggestions for grammar and coherence. It was also used for suggestions on how to shorten paragraphs. The output was always thoroughly reviewed and not directly copied to ensure the final content reflects the original thought and analysis.* *Please note:* Content generated by GAI that is used as a source in the paper requires correct use of quotation marks and source referencing. Read the guidelines from Copenhagen University Library at KUnet here.