Natural Language Processing Project, Block 1, 2023

Marcus Thomsen (HVJ727) **Kasper Schiller (FGZ829)** **Polina Ziboreva (RLB642)**

1 Week 36

We load the Answerable TyDi QA data (Clark et al., 2020) and create a dictionary that holds training and validation data for the three languages. In addition, before completing any tasks in this project, we have removed punctuation and stop words from the data.

(a) We have computed some statistics for respectively the training and validation data sets and translated the results to English as seen in Table 1.

(b) The 5 most common words for the three languages are reported in Table 2. The overlap between the top 5 words in a document's plain text and the top 5 words in a question suggests that the answer to a question may often contain a word present in the question itself.

(c) We have implemented a *Oracle* function which labels to the data, marking an empty answer text as 0, otherwise 1. We have created a rule-based classifier, which has the following logic: Return 1 if the amount of common words in document_text and question_text is larger than or equal to a non-negative self-chosen integer n. The validation data is passed to a function using our rule, where n=2 so that 2 or more words from document and question should be equal. The results are shown in Table 3.

Language	Dataset	No. of Q	W/Q	Answerable Ratio	W in Q	Unique W
Indonesian	Train Set	11394	3.37	0.5	38374	5558
	Validation Set	1191	3.62	0.5	4306	1285
Bengali	Train Set	4779	7.04	0.5	33623	3744
	Validation Set	224	7.43	0.5	1664	431
Arabic	Train Set	29598	4.12	0.5	121969	16183
	Validation Set	1902	3.97	0.5	7556	2208

Table 1: Basic statistics for the data set. Here, Q = Questions and W = Words.

	Document		Question	
Arabic	classification	15655	child	1564
	general	13640	Located	1540
	1	8364	It was completed	1334
	son	6161	general	1159
	a	5482	city	1002
Bengali	is	2916	name	837
	in	2365	in	522
	do	2355	where	300
	as	1334	Bangladesh	266
	by doing	1293	born	252
Indonesian	Indonesia	2371	wide	858
	1	2370	Name	736
	own	1802	time	538
	country	1702	Where	508
	2	1679	Indonesia	470

Table 2: The 5 most common words in the documents and questions from the training set and their frequency.

	Arabic	**Bengali**	**Indonesian**
Accuracy	66.3%	72.77%	68.77%

Table 3: Accuracy of the 3 languages using the rule-based classifier

2 Week 37

We have implemented three different language models. To compare the models we used perplexity, which indicates how well the model predicts a given text, with lower values indicating better predictions. The results are shown in Table 4. We see that in general the Bigram model performs better, followed by Unigram.

Unigram-model. The Unigram-model assigns word probabilities based on their frequency in the

Model	Language	Q Perplexity	Doc Perplexity
Unigram	Indonesian	216	2337
Bigram	Indonesian	170	1873
RNN	Indonesian	1253	2493
Unigram	Bengali	242	1442
Bigram	Bengali	131	1126
RNN	Bengali	1413	3391
Unigram	Arabic	536	6606
Bigram	Arabic	369	4891
RNN	Arabic	760	2391

Table 4: Perplexity for the different models

text's vocabulary. To handle out-of-vocabulary words, we introduced "00V" injection, replacing new words with "00V." This improved the model's performance, but it's important to note that more than 1/8 of words are substituted with "00V," resulting in a low perplexity score. In conclusion, the high likelihood of outputting "00V" for many words is a problem.

Bigram-model. This model analyzes pairs of adjacent words, but it can overlook vital context if a word's meaning relies on non-adjacent words. Therefore, we implemented Bigram using our Unigram-model for smoothing, resulting in improved perplexity compared to the Unigram-model. By analyzing word-pairs the model has a better understanding of language details.

LSTM-model. Long short-term memory network is an extension of RNN. But unlike RNN they do not suffer from vanishing gradients, which for longer sequences can be a problem. Thus, we chose to build a LSTM-network as our third model, since we believed that it would be the better model for document_plaintext due to the longer sequence length. We used the code from Lab 6 for the model. We were not able to optimize the hyperparameters for the model due to the long training time. Instead, we chose the following parameters: `lr = 0.0001, n_epochs = 4, lstm_dim = 2048, lstm_layers = 4, batch_size = 126, seq_len = 126`. We do acknowledge that using the same hyperparameters for both question_text and document_plaintext is suboptimal due the difference in structure, as well as using the same hyperparameters for different languages.

3 Week 38

We have implemented three binary classifiers to predict whether a question is answerable or not. Accuracy results are shown in Table 5.

	LogReg (GloVe)	LogReg (BPEmb)	Transformer (BERT)
Indonesian	69%	71%	89%
Bengali	65%	70%	83%
Arabic	76%	74%	94%

Table 5: Accuracy for each Binary Classifier

Logistic Regression (GloVe). For this model, we utilized the gensim library which provides pre-trained word vectors. Since library does not have models for our languages, we have translated the data to English using the *googletrans* library. Then, we utilized the pre-trained model *glove-wiki-gigaword-100* which provides pre-trained 100-word vectors (Pennington et al., 2014). We perform regression with the classic logistic function, where our features x_1 and x_2 are respectively the question text and document plain texts word vectors. The target label y is whether a question is answerable or not. The model was trained and validated on respectively the training and validation set. It is seen that the Arabic model, with an accuracy of 76% is superior. This could be due to two factors. First, the Arabic data may be more linear separable, allowing logistic regression to create a straightforward classification rule. Secondly, the translation process itself might play a role; it is possible that the translation library has greater proficiency with Arabic.

Logistic Regression (BPEmb Tokenizer). For this task, we have used Byte-Pair Encoding embeddings, which the words from a language into smaller parts, and learns embeddings from these parts. The model contains pre-trained embeddings for the 3 languages (Heinzerling and Strube, 2018). We utilize the same logistic function as before but with only one feature, x_1, since we concatenated the question text and document plain text with the separator *[SEP]*. The results are very closely aligned to those obtained with the prior model. Only the Bengali case, has a significant different performance; here we have an increase in accuracy of 5 percent points. Three possible reasons for this difference are considered. It could be that the translation to English of the language in the prior model affected it too much. Or, that the BPEmb embedding simply just works better than the Glove

embedding. Or, that the $[SEP]$ method works better than having two seperate features.

Transformer model (BERT). For this part we used the guidelines of the Huggingface Sequence Classification guide. But instead of trying to classify the sentiment, we now tried to classify whether a question was answerable or not. We chose to use the pretrained *bert-base-multilingual-cased* model (Devlin et al., 2018). We then fine-tuned the model for the task of classification of answerability. To do so, we concatenated the question_text and the document_plaintext into a single string that was then tokenized and used as input. We did try to use [SEP] between the two text pieces, but it did not give any improvement, this might be due to the fact that at the end of each question we always have a question mark, which then in itself works as a separator. The model showed great results, with the classification of Arabic being the best. This is likely due to the fact that the BERT model has a lot of pretraining on Arabic and further that the dataset for Arabic is larger than the two other languages.

4 Week 39

Two Transformer based QAs. Transformers have become more and more prevalent architecture compared to recurrent neural networks (RNNs). This is partly due to challenges such as vanishing gradient, which RNNs suffers from. Transformers circumvent this issue with self-attention mechanisms, allowing the model to weigh the importance of different parts of the input regardless of their position. In our work, transfer learning is utilized to fine-tune pre-existing models, adapting them to our specific dataset and task requirements. The specific models used here are multilingual Bert for the first model and language specific transformers for the second. To do all of this we followed the coding from Lab 6, but instead of using SQuAD metric we used the SQuAD_v2 metric and tailored our dataset to the SQuAD v2 format, such that we could get the corresponding EM and F1 for each model for each language. Thus we fine-tuned the models to predict the start and end tokens and from this deducing the answer span. For the multilingual model we used *bert-base-multilingual-cased* (Devlin et al., 2018), while we found similar models for the monolingual cases [1].

IOB-tagging. IOB-tagging is a sequence labeling technique used to indicate the beginning (B), inside (I), and outside (O) of entities in text. In our data-set we are working with two entities, "answer" and "not answer". To perform the method, a function "iob" was implemented. The function returns the corresponding tags for each token in the document_plaintext.

To make the model we used the pre-trained *bert-base-multilingual-cased* model (Devlin et al., 2018). This model was used on all 3 languages. The models was trained using PyTorch's Dataloader class and the AdamW optimizer using 1 epoch. Then, the models were evaluated on the validation set.

Results. As expected, using the same training architecture and the same hyperparameters we see in Table 6 that using the language specific BERT transformers to give us improved performance for all three languages. This is even though BERT already includes all three languages. But for the momonolingual BERT has been further trained specifically on each of the three languages and thus improves the performance quite significantly. Bengali language has the lowest scores, this could be due to the vocabulary size. The IOB Tagging System is definitely the superior model here.

Model	Language	EM/ F1
SE_multilingual	Indonesian	35.01/41.56
SE_monolingual	Indonesian	44.25/49.61
IOB tagging system	Indonesian	58.52/54.63
SE_multilingual	Bengali	25.45/32.12
SE_monolingual	Bengali	33.48/38.35
IOB tagging system	Bengali	52.23/49.74
SE_multilingual	Arabic	31.70/40.40
SE_monolingual	Arabic	36.49/45.15
IOB tagging system	Arabic	63.93/58.4

Table 6: Table of performance for the QA models.

5 Week 40

For this part we utilized the binary classification transformer from week 38 and the BERT sequence classifier from week 39 which predicts the start and end token of the answer. We chose two different models using the same transformer to see

[1] Indonesian model: *indolem/indobert-base-uncased*, Bengali model: *csebuetnlp/banglishbert*, Arabic model: *aubmindlab/bert-base-arabertv02*.

if we could see any significant difference in the attention between the two. We looked at the attention layer 12, since it is the last layer in the BERT transformer, and thus the most interpretable. We made an attention matrix, but found the output to be very difficult to interpreted. Thus, we chose to use the function `model_view` from the library `bertvix`, which gives a greater visualization as seen in Figure 1 for Arabic. The rest of the interactive part can be found in the Appendix.

Binary Classification. Looking at the attention in layer 12 it is difficult to make any conclusion on the model predictions based on the visualization. Although it does seem to look as though the attention goes towards the [SEP] at the end of the sentence whenever the question is answerable. Whether this is a coincidence of the few instances we have to look at or not is difficult to say.

Sequence Labeller. Here we would have hoped for the last layer to show clear attention towards the answer itself. For some of the languages, it seems as this is true. A great example of this is the sequence labeler for Arabic language. This can be seen in Figure 1 where the attention for a lot of heads is towards the actual answer. But this is only really clear for Arabic, and might not be for all instances. With a more precise model, we believe that this would become far more obvious and clear. But with the limited EM of our models, it becomes quite vague.

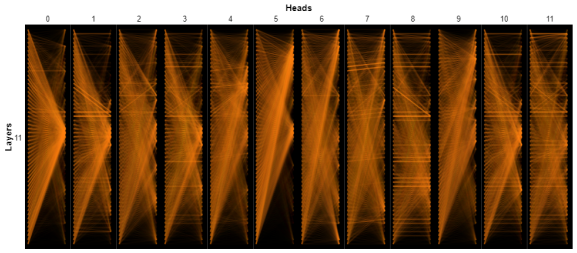

Figure 1: ARAB_SC

6 Week 41

Binary Classifier. For this part of the task, we utilized our binary classifier model from Week 38, that is the Logistic Regression with the BPEmb tokenizer (one for each language). We start out by tokenizing both the training and test data for each language, using the seleted BPEmb model for the language. Then, we train the classifier on one language and test it on the other two languages. We do this for each language and obtain the results seen in Table 7. Comparing the results to what we obtained last week, we see that the all of the cases of the model has worse performance when trained on a different language; this makes sense, the BPEmb model may have limitations when dealing with languages other than the one it was originally trained on. To solve this problem we could use a more multilingual model such as XLM-R (Reimers and Gurevych, 2019). There is no pattern in terms of which language is the better to train on, but the language achieveing the highest accuracy when tested on is the Bengali language by far. It is the only language with an accuracy significantly larger than 50%.

Training / Testing	Indonesian	Bengali	Arabic
Indonesian	71%	**58%**	**50%**
Bengali	**50%**	70%	**51%**
Arabic	**50%**	**61%**	74%

Table 7: Logistic Regression with BPEmb Tokenizers

Sequence Labeller. For this part we used the IOB-tagging method from week 39. We started off by tokenizing the training and validation data for three languages. Using PyTorch for data batching, we trained the model on each language and tested it against the validation sets of two other languages. The results are shown in Table 8. The results show that using Arabic language to train the model yields better predictions in average. Using Indonesian language for training is really efficient for prediction on the Arabic test-set.

Training / Testing	Indonesian	Bengali	Arabic
Indonesian	54.63	**36.38**	**58.58**
Bengali	**29.96**	49.74	**29.28**
Arabic	**47.4**	**44.35**	58.4

Table 8: F1-Score | Sequence Labeller (IOB)

7 Conclusion

We conclude that we were able to implement, fine-tune and evaluate several NLP models introduced in the course on the TyDi QA dataset with acceptable results. They are all described in this report.

References

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*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Benjamin Heinzerling and Michael Strube. 2018. BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Appendix

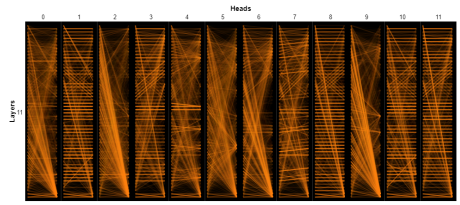

Figure 2: ARAB_QA

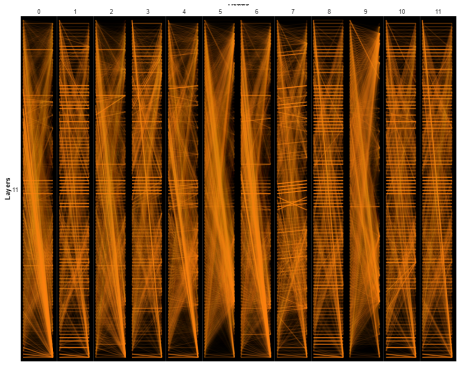

Figure 3: BENG_QA

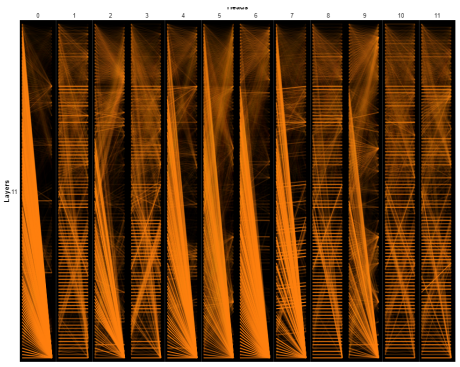

Figure 4: BENG_SC

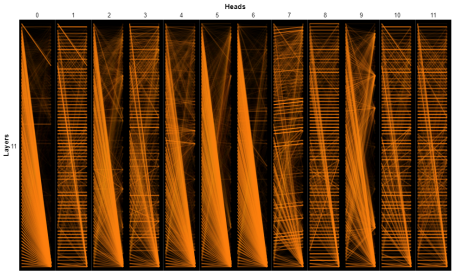

Figure 5: INDO_QA

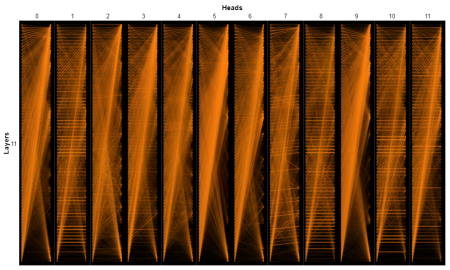

Figure 6: INDO_SC