NLP Project

Group name: RCDeportivo
Víctor Escudero González, Carolina Branas Soria, Christian Turner
3rd November 2023

1 Week 36

1.1 Part 1: Importing Dataset and Summarizing Info (Carolina)

We converted the train and validation datasets into dataframes and filtered rows by language. We initially thought of translating all corpuses to English and then work on them using specific English tokenizers and embeddings, as English resources are widely used. Due to the high time complexity of translating each corpus, we instead decided to treat each language as a unit and use specific embeddings and tokenizers for each. Table 1 shows the general statistics and class distribution of the datasets.

1.2 Part 2: Word Count for Each Language (Víctor)

For the three languages, we have found the 5 most common words for both documents and questions, and then translated them into English to see their meaning. Table 2 shows the most common words with their occurences. Although we applied punctuation removal, some language punctuation still appeared in the results, such as the Arabic question mark or the Indonesian full stop. Apart from that, we can see that the documents mainly have stopwords as their most common words, like prepositions or conjuctions, wheres in the questions the most common words are the question opening words (How much, What, When, etc.).

1.3 Part 3: Rule-based classifier (Víctor)

For this part, we have make a rule-based classifier that served us as a baseline for the models in the followed weeks. In this case, the rule-based classifier used the most common words from and also the mean lengths in each document for answerable and non-answerable samples over the training set, and had **two rules**: if 10% of the words are among the first 1000 most common or if the length of the document is larger than the mean of answerable documents, then return answerable, otherwise return non-answerable.

Table 3 contains the accuracy (using the oracle function) on the validation set. We got a bit better results than predicting randomly, therefore, it seems that data is too complex to be predicted by a rule-based classifier.

2 Week 37

2.1 Bigram (Víctor)

For implementing a Bigram Language Model, first, we need to get the vocabulary of our training data. In order to do that, I have added each word to the vocabulary the first time I saw it, and after, substitute them in the document by an **OOV** (Out-Of-Vocabulary) token. This allows us to get an estimation of how often a new word appears in the documents. Having the vocabulary built with the training set, I replaced the unseen tokens in the validation corpus by OOV.

To avoid 0 count of a history to be seen, and therefore, undefined probability, I used **Laplace smoothing**. With Laplace smoothing we can move mass probability from overestimated values to underestimated. I applied Laplace smoothing with smooth parameter $\alpha = 0.1$

The code in the notebook was mostly taken from the course's GitHub, adapting it to our datasets. For tokenizing, I used the library *polyglot* [1], which has tokenizers for many languages. I used **perplexity** as evaluation measure over the validation set, since it shows how uncertain is a model in its prediction. Table 4 shows the perplexity along with the vocabulary size. As we can see, the perplexity is proportional to the size of the vocabulary.

2.2 RNN (Carolina)

Recurrent Neural Networks (RNNs) can take into account context better than other simpler algorithms like N-Gram. We decided to experiment with an initial RNN that can later develop and become a more complex algorithm, like, for example, LSTM.

For feeding the RNN, we used the questions and the documents corpuses for each language. As RNNs have a high time complexity, some corpuses were subsampled. The corpuses were initially preprocessed to eliminate the annotations containing brackets, as we initially proved that these were noisy and induced the model to predict these particular tokens continuously, hence decreasing performance. For each language, we built a **vocabulary** of unique words using both the training and validation sets, and an embedding matrix with the size of the vocabulary and 100 dimensions using the **BPEmb** model[2], plus a set of **OOV** words. We turned the input text into a vector of indices. Each index corresponds to a word in the vocabulary, so words can be mapped to an index in the total vocabulary when obtaining the output probabilities from the model. Additionally, we padded the input vectors with 0s so they have the length of the longest input text in either the validation or the train corpuses.

Each feature was split into **inputs** of four tokens (n=4) and **targets** (n+1). This way, the RNN can be fed more context than just the previous word to build the predictions, improving the results in comparison to just using the previous word as input. The features were turned into tensors and fed into a simple RNN (batch size = 64, 100 layers, output = vocabulary size, learning rate = 0.001) that **outputs probabilities** for each word in the vocabulary, choosing the highest probability as the final prediction. The model was run at 10 epochs for the questions corpus and at 5 epochs for the documents' text corpus, and predictions were evaluated by calculating both **cross entropy loss** and **perplexity**.

As we initially expected, due to its simplicity, the model did not capture the context of the input vectors accurately, scoring very poorly on perplexity and cross entropy loss (see Table 5 and Table 6). In addition, subsetting the corpuses reduced the richness of the vocabulary, mining performance. We also observed that the classes were not distributed equally: for instance, as questions naturally contained a high amount of “?" tokens, the model predicted this token continuously. On the other hand, it is an ongoing discussion to remove or not specific characters for language modelling tasks, as these characters are needed to build richer predictions.

2.3 LSTM (Christian)

RNN models often struggle to make use of information distant from the current point of processing. To address this issue, an LSTM has been used which aims to enhance long-term dependency handling with context layers and specialized gates to improve performance.

The data was initially divided into training and validation sets, further segmented by language, and structured into separate dataframes for questions and document texts. To prepare the data for the model, we employed the multilingual BPEmb library with a vocabulary size of 320,000 and 100 dimensions. Unique training vocabularies were created for each language from the training and validation datasets, with out-of-vocabulary tokens and end-of-sentence markers added.

To feed the data into the model, we created index vectors to access the embedding matrix. The tokenized data was transformed into PyTorch tensors and batched for efficient computation. The LSTM model consists of 2 layers running for 5 epochs, with gradient clipping used to mitigate gradient issues. Model performance is assessed using cross-entropy loss and perplexity.

Comparing the RNN and LSTM is not straightforward since they use different vocabularies. Ideally, a common vocabulary would be used for a fair comparison, but time and computational constraints prevented such synchronization. Evaluation metrics and visualizations of loss for these models can be found in the appendix. [12 13 14 15 16 17]

3 Week 38

3.1 Logistic Regression with BPEmb and TF-IDF Features (Carolina)

BPEmb[2] is a collection of pre-trained embeddings in up to 275 languages using Wikipedia corpuses. Due to this large language support as well as the suitability of the corpus in which the embeddigns are trained on, the BPEmb embeddings were a reasonable choice to train a simple, initial logistic regression model. The chosen vocabulary size for all of them was 25000 words, and embeddings of 100 dimensions in every language.

The preprocessed document texts and questions (lowercased, no punctuation or blank spaces) were the input to the classifier. This preprocessing slightly improved our results, as it helped **standardize the data and reduce noise**. This input was then embedded using the previously loaded BPEmb model, obtaining word vector representations that were averaged to get a suitable input for the regression. We performed **PCA**, retaining 80% of the **variance** in the original data, to also **reduce the dimensionality** of the input.

We also used **Term Frequency-Inverse Document Frequency** (TF-IDF) to obtain features that can be fed into the logistic regression. TF-IDF is a technique that vectorizes words based on their importance in a document in relation to the entire corpus. It can “stretch the word count as well as “compress” it. In other words, it makes some counts bigger, and others close to zero.”[7]. We obtained many features that were close to 0, increasing **sparsity** and making it impossible to reproduce PCA as in the previous case. To solve this, we performed truncated SVD with 10 components instead. In both cases, we performed hyperparameter tuning to improve performance, mostly taken from the course’s Github.

We suspect that the overall low accuracy is due to the impossibility of the embeddings to provide enough information to the model. In the end, we are reducing the richness of a whole language to a set of vectors, and in this process, a lot of information is missed. We suspect that one reason BPEmb embeddings performed better than TF-IDF might be that they capture **subword information** better, which is beneficial for morphologically complex languages like Arabic, Bengali or Indonesian. Refer to Table 7, Table 8, Table 9 and Table 10 for the results.

3.2 LSTM (Víctor)

For the sequence classification part, I decided to implement an **LSTM** based model. One of the main characteristics of an LSTM is that it can achieve better results than RNNs when it regards to long sequences and remembering parts of the sequence that appear distant in the document. First, I tokenized documents and questions by using the library *polyglot*, which has several features for many languages. In this preprocessing step, I discarded the texts that polyglot detected as a different language. Having the tokens, I applied **lowercase**, and removed **punctuation** and **stopwords**, since these tokens do not give much information to the documents. After these preprocessing steps, I converted the tokens into numerical vectors. Polyglot also contains 64 dimensional word embeddings trained from wikipedia texts. For converting each document into vectors, I used the average of each token representation. For the model training part, to give the same importance to both questions and inputs, I used two separated LSTMs, followed by a linear layer of output size 2, since it is a binary classification, and a softmax function to convert these logits into probabilities. For the loss, I chose the **Binary Cross-Entropy Loss**, and the Adam optimizer. I trained each model on 50 epochs with a batch size of 32. To find the most suitable number of hidden sizes and number of layers, I used 4-fold **cross-validation** on the Bengali training data on 16 and 32 as hidden size parameters, and 1, 2 or 3 layers. I ended up with a **hidden size** of 32 and 1 **layer**.

As evaluation measure, I chose **accuracy**, **F1-Score**, precision and **recall**. Table 11 shows the evaluation measures in the validation set. Arabic is slightly better than the rest overall, might be due to the fact that data is bigger in this case. Bengali tends to predict more 1s, due to its recall value, than the rest, but with way less data, its results are even better than Indonesian and similar to Arabic, this also may be related to the fact that cross-validation was performed on the Bengali training data, so the model is more suited for Bengali.

3.3 Transformer Model (Christian)

For the task of supervised classification we have chosen to fine-tune the transformer model *xlm-roberta-base*. Implementing a transformer model addresses one of the biggest shortcomings of RNN and LSTM models – attention. The model takes input as batches of tokens which it processes in parallel to create embeddings, and then it applies self attention to capture the relationship between the tokens. Unlike the basic RoBERTa model, this version of the model has the advantage of being multilingual.

The data has been split into training and test sets, and once again split by language. For computational ease the training and validation sets have been subset to a length of 1000 and 224, respectively. To ensure the tokens in our data have corresponding pretrained embeddings, the xlm-roberta tokenizer was used. The parallel nature of the transformer model necessitates that the question and document texts be concatenated and split by special tokens. Padding was set to the max length of inputs (128) and truncation set to 'only_second'. In preparation for the data to be fed to the model, the input ids, attention masks, and ground truth labels for both the training and validation data were converted to PyTorch tensors. For the optimizer and loss function, AdamW and cross-entropy loss were used, respectively.

Given the processing power needed for this model, only 1 epoch was run for each language. The fine-tuned models were then saved and tested on the validation datasets. The evaluation metrics of precision, recall, and f1 score were used and are available in the appendix, table 17. To increase the model's performance, hyperparameter optimization via grid search was considered but rejected due to processing limitations.

4 Week 39

4.1 Transformer: distilbert-base-multilingual-cased (Carolina)

We used HuggingFace's library *transformers*[6] to perform span-based question answering using the provided dataset. The first model choice was the *'distilbert-base-multilingual-cased'* pretrained model, a smaller version of the *bert-base-multilingual-cased*, trained in 104 languages on Wikipedia data, including the pertinent ones for this project. Unlike BERT, the distilled version only counts with six encoder layers. This model is beneficial due to its multilingual nature and its reduced time complexity in comparison to BERT.

For **preprocessing**, we concatenated the questions and contexts, with padding to the maximum length (set to 384), and tokenized them using the specific model tokenizer. We manually stated that, if the answer was not contained within the context, the labeled answer start and end were set to 0s. Due to the lack of computational resources and time, the train and validation datasets had to be subsampled (600 and 200 rows respectively) for the three languages. Subsetting might have reduced the accuracy of the model significantly, and with more time and resources, we would have reported the results using the whole dataset.

The transformer was **fine-tuned** over 5 epochs on the training set using the pre-built Hugging Face **Trainer**. **F1-Score** was chosen as the evaluation metric, as **exact-match** highly penalizes predictions without considering any shared tokens. We compared the F1 scores between the original (Base F1 Score in Table 12) and the fine-tuned model, (F1 After Fine-Tuning in Table 12) and despite being quite low, the F1 score improved when fine-tuning the model. One reason of having such low performance could be that the context of the documents is large, hence making it difficult for the model to know what to pay attention to, in addition to subsetting.

4.2 Transformer: bert-base-multilingual-cased (Víctor)

In this week, we use HuggingFace's library *transformers*[6] to create a **span-based question answering** model taking into account the answer_start and answer_end features of our dataset. For the case,

In my case, I used *bert-base-multilingual-cased* pretrained model, which is a multilingual version of Bert that supports up to 104 different languages.

To preprocess the data, HuggingFace also has different tokenizers implemented, that give us several options, such us applying padding to the tokens batch, separate token input into several parts if it is too large, etc. In my case, I applied padding and passed the attention mask to the model input. Apart from tokenizing, I also needed to apply some preprocessing to the labels: when the answer is not answerable, I labeled the answer start, and the answer end as both 0s.

I fine-tuned the transformers over **3 epochs** on the training set, and I chose as evaluation measures **exact-match** and **F1-Score** on the validation set. Exact-match measures the percentage of spans predicted that were exactly the same to the ground truth. Since now we are predicting spans, the implementation of F1-Score is a bit different than before: in my case, I took as true positive the number of tokens that are shared between predicted and true label, without order. Then, false positives are the count prediction - the shared, and the false negatives are the count of true tokens - the shared. Table 13 shows a comparison on the evaluation metrics between languages. In this case, we observe that the model performs worse on Bengali, where data is smaller.

4.3 Transformer: xlm-roberta-base (Christian)

In an effort to compare the performance of BERT and RoBERTa based models, I implemented a span-based question answering model based on the xlm-roberta-base model, following the processes in section 4.2, adapted from Victor's bert-based-multilingual-cased model.

The basic version of the RoBERTa model surpasses BERT by training on more data, for longer periods, and with bigger batches, resulting in superior performance on GLUE, SQuAD, and RACE benchmark tests [3].

When fine-tuned and evaluated on our dataset we found the following metrics: Arabic - exact-match: 0.71, f1-score: 0.781; Bengali - exact-match: 0.6, f1-score: 0.785; Indonesian - exact-match: 0.67, f1-score: 0.737. Performance on Arabic and Bengali improved with the RoBERTa-based model, but scores for Indonesian declined. This may stem from differences in training data between the models, but these are complex multilingual models and many factors could be at play.

5 Week 40

5.1 Attention: Span Question Answering (Carolina)

Attention mechanisms recently became popular because of their possibility of capturing context. BERT models use self-attention, which "is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence"[4]. We used the library 'bertviz'[5] to dynamically visualize the attention of our span-based question answering transformer **distilbert-base-multilingual-cased** (which contains 6 layers and 12 attention heads) for a sampled question and its context.

We chose the question and its context because, in a good performing transformer, when hovering over specific question tokens, we would expect the attention to be directed to the tokens in the context containing the answer. For the span-based question answering fine-tuned transformer, we observe that, in the three languages, the attention head focuses a lot on the [SEP] and [CLS] tokens (See especially Figures 2, 4, 5). This is called a **SEP focused attention pattern**, and it typically happens when the model cannot discriminate which tokens to focus on. Because of this lack of discriminating which tokens to pay attention to, the model poorly captures any context. Due to the low accuracy of the results, we expected something like this before visualizing the attention.

5.2 Attention: Binary Classification (Christian)

When interpreting the visualized self-attention of the xlm-roberta-base model used for binary classification, the task of the model and available information must be taken into consideration. The aim of our model is to predict if a question, given a document text, is answerable. Patterns which lead to a prediction may be found in various sections of the models input, so we must not look for attention to be solely focused on the answer tokens.

To give visualizations the best chance of being interpretable, instances which were correctly predicted to be answerable were chosen. Further, the shortest such instances were chosen for ease of visualization. The model consists of 12 layers and 12 attention heads. When each of the validation instances are viewed from the head view, the attention at layer 0 appears to be fairly randomly distributed and roughly seems to focus on the special tokens as it progresses through to layer 12 [6-11]. This pattern does not provide any insight into how the model arrived at the prediction that the question is answerable; however, unlike in the previous model described in section 5.1, this model performs fairly well with f1-scores in the upper 0.70s to low 0.80s, so there must be an indiscernible pattern detected by the model.

6 Week 41 (Víctor)

6.1 Classifier zero-shot evaluation

In this section, I chose the **xlm-roberta-base** fine-tuned model made by Christian on week 38 [3.3], and tested the models trained on each language against the other two. The results are shown on table 14. Results are very surprising, since they are very close to the monolingual results. That could be explained with the fact that the model was fine-tuned for only one epoch on a dataset subset. This shows that transformers models like xlm-roberta-base can have really good metrics on zero-shot evaluation.

6.2 Sequence labelling zero-shot evaluation

For sequence labelling cross-lingual evaluation, I have chosen the bert-base-multilingual-cased fine-tuned on week 39 [4.2]. Table 15 shows the cross-lingual evaluation results. In this case, results are slightly worse than on the monolingual evaluation, but as in the classifier case, we still get pretty strong results, taking into account that exact-match gives a big penalty on predicted sequences that are almost similar.

References

[1] Yanqing Chen and Steven Skiena. 2014. Building sentiment lexicons for all major languages. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pages 383–389.

[2] 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).

[3] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

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

[5] Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy. Association for Computational Linguistics.

[6] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

[7] Alice Zheng and Amanda Casari. 2018. *Feature Engineering for Machine Learning*. O'Reilly Media, Inc.

7 AI Assistance

Chat GPT-3.5 Version was used as a code generator assistance tool for some of the code contained in the notebooks, as well as for guidance with overall workflow for some of the models.

Language	**Train Size**	**Validation Size**	**Answered Questions**	**Unanswered Questions**
Arabic	29598	1902	50%	50%
Bengali	4479	224	50%	50%
Indonesian	11394	1191	50%	50%

Table 1: General Statistics

Column	Language	Common words	Occurences in corpus
	Arabic	،, in, from, on, to	105915, 89735, 61716, 28071, 22080 times
Documents	Bengali	।, O, And, Is, By	26347, 5202, 5102, 4454, 3657 times
	Indonesian	Which, and, Of, on, from	24424, 23987, 17572, 11025, 10066 times
	Arabic	؟, what, when, he, from	29572, 7455, 7130, 6762, 6309 times
Questions	Bengali	Key, Name, How much, Is, Any	940, 837, 802, 800, 764 times
	Indonesian	is, When, Which, what, First	2007, 1843, 1822, 1774, 1132 times

Table 2: Common words for document and question in the training set

Language	accuracy
Indonesian	0.573
Arabic	0.534
Bengali	0.611

Table 3: Accuracy of rule-based classifier on the validation set

	Language	Vocabulary size	Perplexity
	arabic	200279	234.083
Document	bengali	50392	121.04
	indonesian	68471	176.94
	arabic	14256	63.006
Question	bengali	3754	26.48
	indonesian	5774	26.93

Table 4: Perplexity for the Bigram model over questions and documents in different languages

Language	Subset	Vocab size	OOV	PPL
Arabic	3000, 600	5649	81.5%	2078.79
Bengali	No	3749	69.1%	2007.99
Indonesian	No	6686	85.4%	6687.73

Table 5: Statistics of the RNN language model taking questions as features

Lang	Subset	Vocab size	OOV	PPL
Arabic	200, 40	8482	83.9%	3120.97
Bengali	200, 40	6438	72.5%	5924.83
Indonesian	200, 40	7025	86.7%	6740

Table 6: Statistics of the RNN language model taking documents as features

Embedding	Language	Accuracy
	Arabic	62%
bpemb	Bengali	63%
	Indonesian	59%
	Arabic	54%
tf-idf	Bengali	50%
	Indonesian	50%

Table 7: Accuracy Metrics for Logistic Regression in the three languages

Embedding	Class	Precision	Recall	F1-Score
	0	0.64	0.59	0.61
bpemb	1	0.62	0.66	0.64
	0	0.53	0.84	0.65
tf-idf	1	0.60	0.25	0.35

Table 8: Classification Metrics for Logistic Regression in Arabic

Embedding	Class	Precision	Recall	F1-Score
	0	0.67	0.52	0.59
bpemb	1	0.61	0.75	0.67
	0	0.50	0.99	0.66
tf-idf	1	0.50	0.01	0.02

Table 9: Classification Metrics for Logistic Regression in Bengali

Embedding	Class	Precision	Recall	F1-Score
	0	0.61	0.50	0.55
bpemb	1	0.58	0.68	0.63
	0	0.87	0.63	0.67
tf-idf	1	0.13	0.21	0.67

Table 10: Classification Metrics for Logistic Regression in Indonesian

Language	accuracy	f1	precision	recall
Indonesian	0.668	0.683	0.704	0.663
Arabic	0.739	0.742	0.738	0.747
Bengali	0.736	0.758	0.711	0.813

Table 11: Evaluation measures on LSTM Classifier on the validation data

Language	**Subset**	**F1 After Fine-Tuning**	**Base F1 Score**
Arabic	600, 200	0.14	0.04
Bengali	600, 200	0.78	0.15
Indonesian	600, 200	0.15	0.03

Table 12: Statistics of the fine-tuned distilbert-base-multilingual-cased transformer

Language	Exact-match	F1-Score
Arabic	0.696	0.768
Bengali	0.5357	0.6015
Indonesian	0.7171	0.744

Table 13: Evaluation measures of fine-tuning bert-base-multilingual cased on span QA

Train language	Test language	accuracy	f1	precision	recall
Bengali	Arabic	0.803	0.827	0.738	0.940
	Indonesian	0.792	0.818	0.729	0.931
Indonesian	Arabic	0.819	0.841	0.75	0.958
	Bengali	0.777	0.803	0.718	0.911
Arabic	Indonesian	0.826	0.838	0.785	0.899
	Bengali	0.763	0.782	0.725	0.8482

Table 14: Zero-shot cross-lingual xlm-roberta-base classifier evaluation

Train language	Test language	exact-match	precision	recall	f1
Arabic	Indonesian	0.6	0.652	0.698	0.652
	Bengali	0.486	0.547	0.595	0.549
Indonesian	Arabic	0.627	0.694	0.713	0.684
	Bengali	0.486	0.524	0.5	0.505
Bengali	Arabic	0.497	0.548	0.611	0.548
	Indonesian	0.574	0.618	0.636	0.604

Table 15: Zero-shot cross-lingual span QA model evaluation

Language	**Data**	**Training Loss**	**Validation Perplexity**	**Vocabulary**
Arabic	questions	4.660	106	5135
	document	5.867	250	13386
Bengali	questions	6.389	670	2050
	document	6.072	352	5900
Indonesian	questions	4.441	105	3374
	document	6.111	439	21957

Table 16: Evaluation Metrics for W37 LSTM Language Model

Train language	accuracy	f1	precision	recall
Arabic	0.8125	0.8320	0.7324	0.9630
Bengali	0.7411	0.7803	0.6776	0.9196
Indonesian	0.8170	0.8392	0.7810	0.9068

Table 17: Binary Classification Evaluation Metrics, xlm-roberta-base

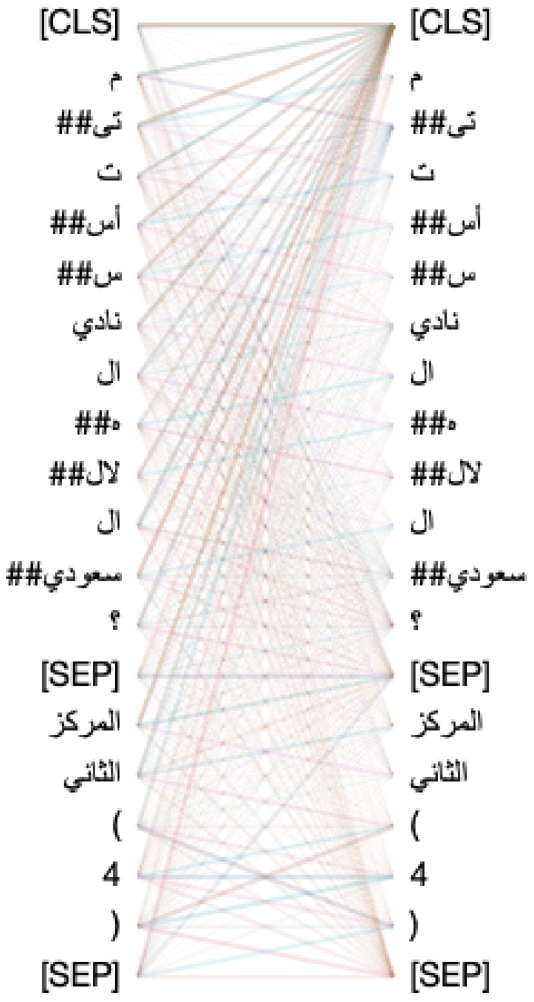

Figure 1: Visualization of layer 0 of the attention for span-based transformer in Arabic

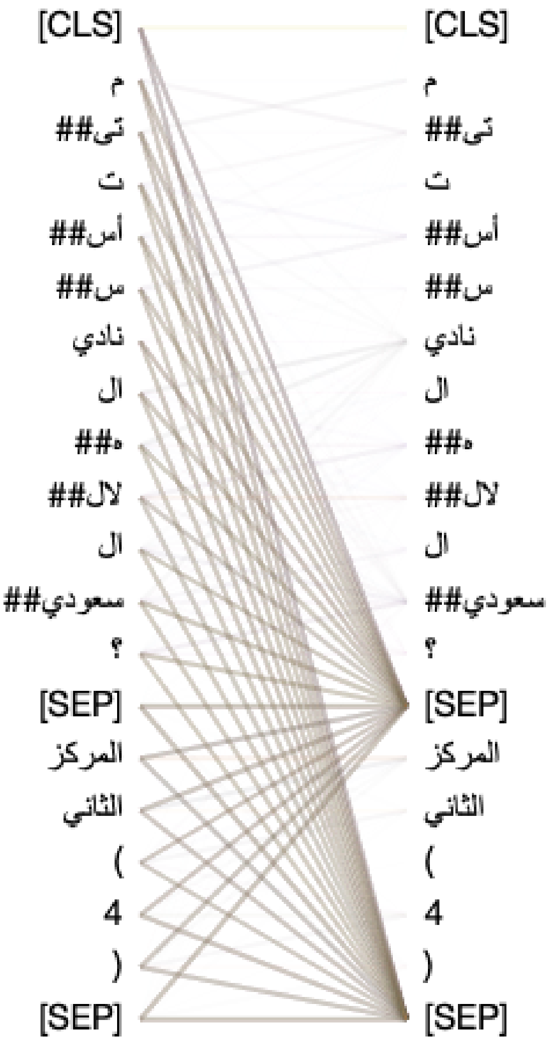

Figure 2: Visualization of layer 5 of the attention for span-based transformer in Arabic

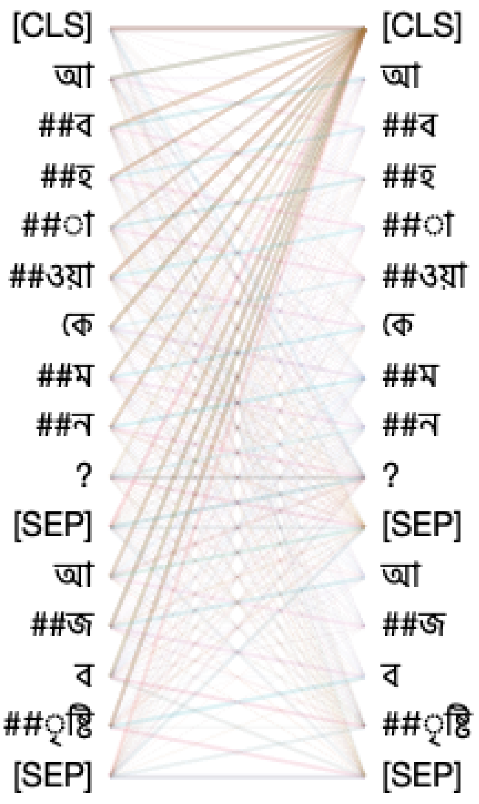

Figure 3: Visualization of layer 0 of the attention for span-based transformer in Bengali

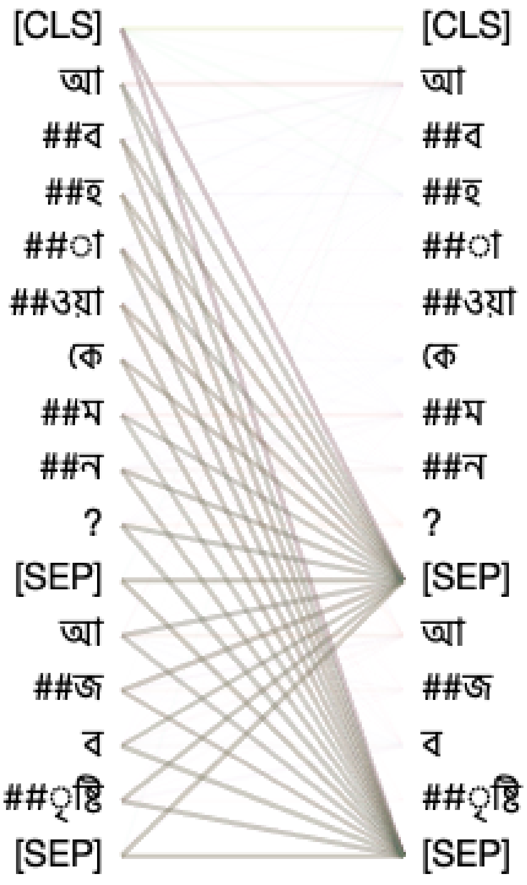

Figure 4: Visualization of layer 5 of the attention for span-based transformer in Bengali

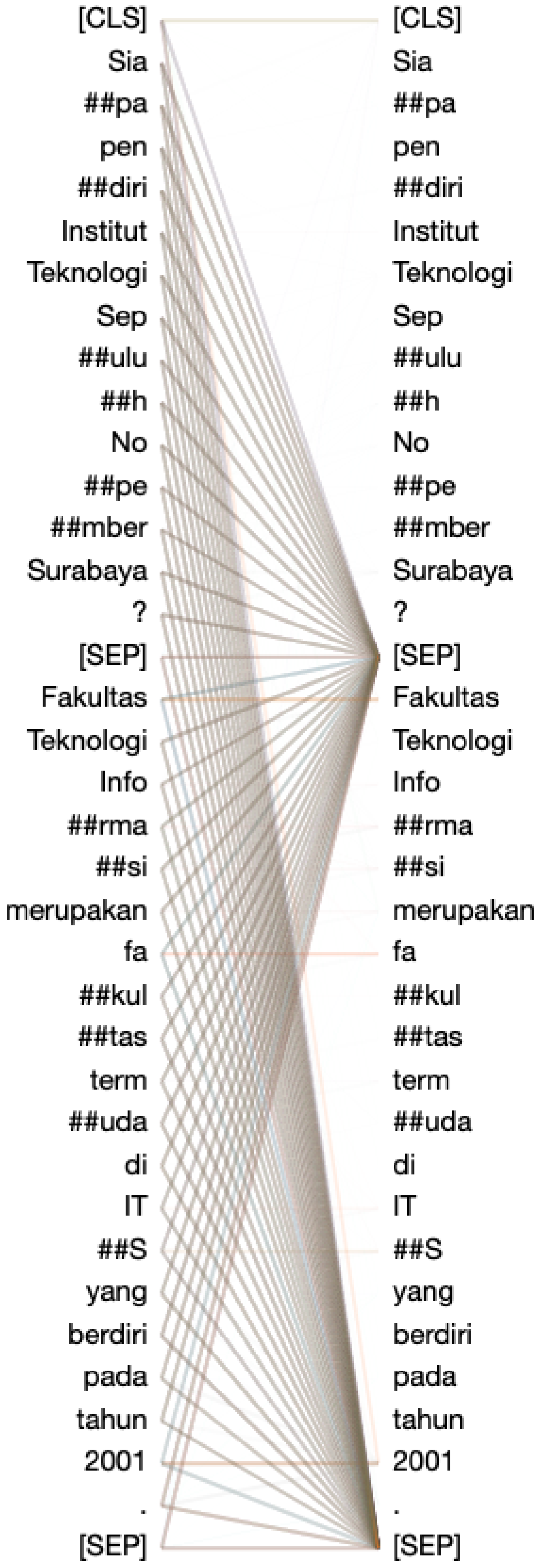

Figure 5: Visualization of layer 5 of the attention for span-based transformer in Indonesian

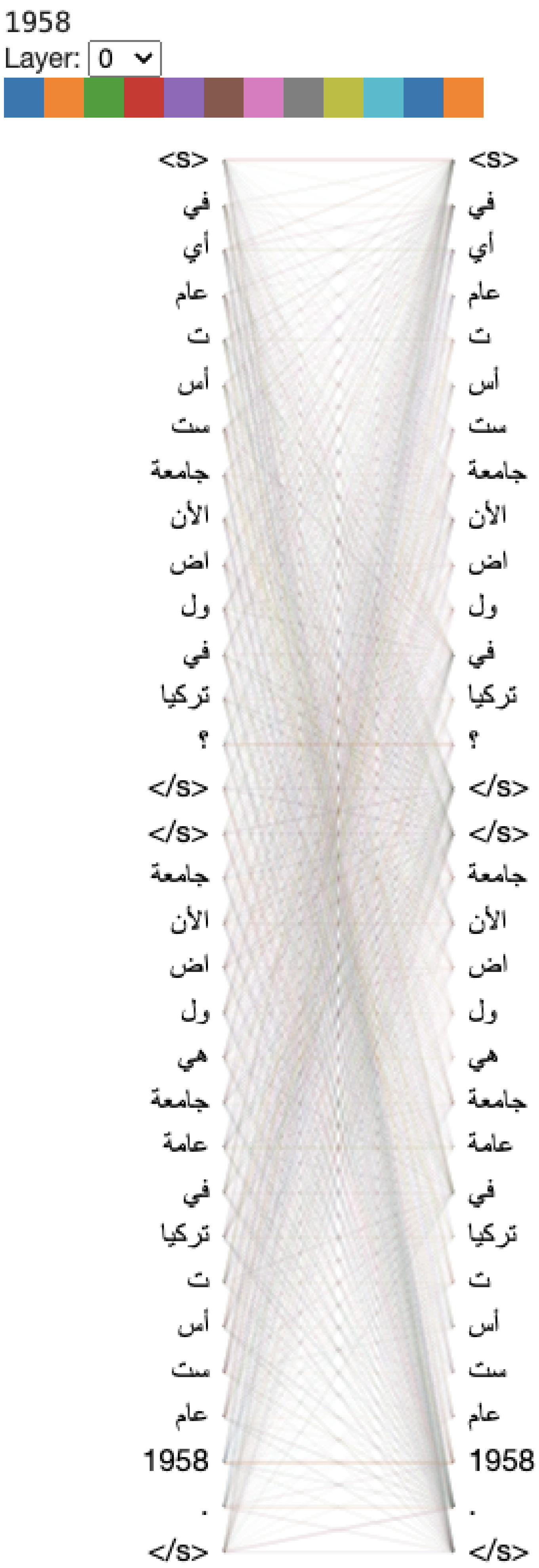

Figure 6: Binary classification attention: Arabic, Layer 0. Answer displayed at top left.

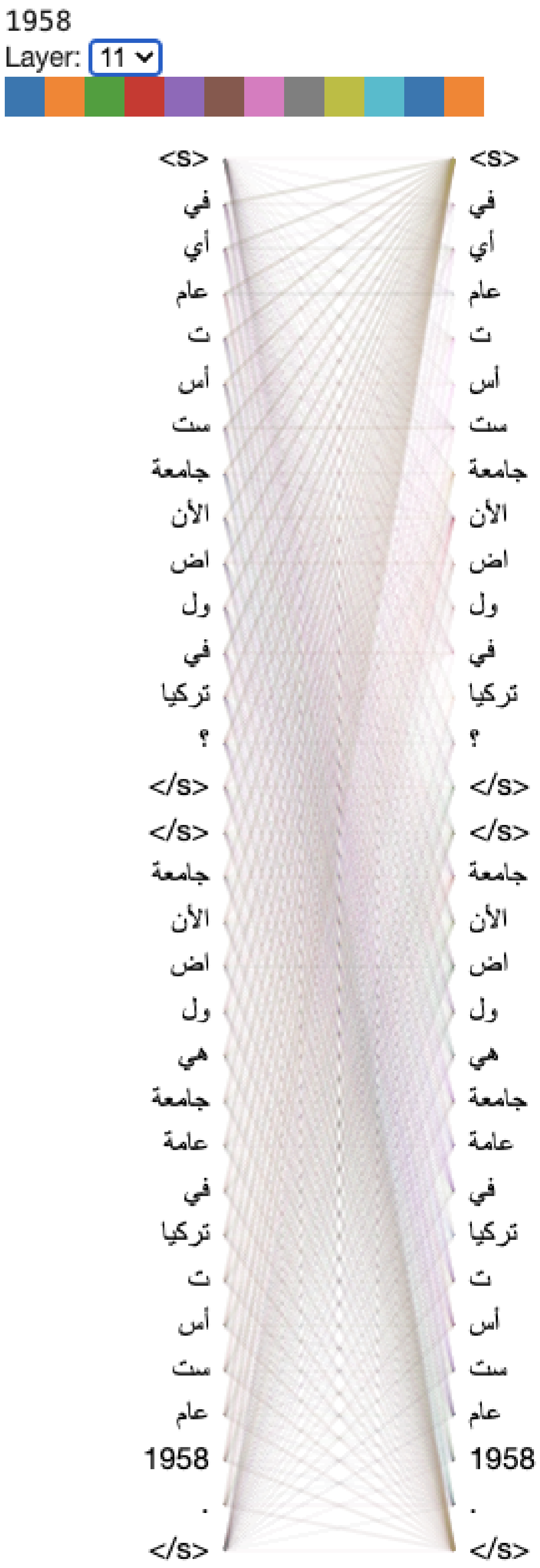

Figure 7: Binary classification attention: Arabic, Layer 11. Answer displayed at top left.

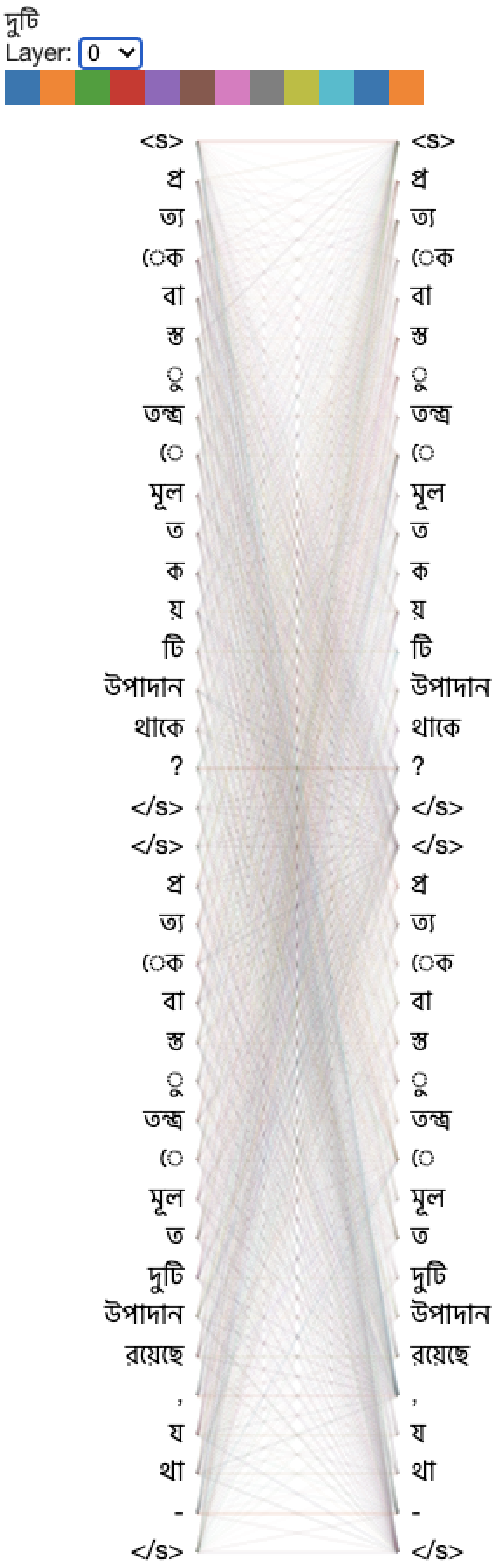

Figure 8: Binary classification attention: Bengali, Layer 0. Answer displayed at top left.

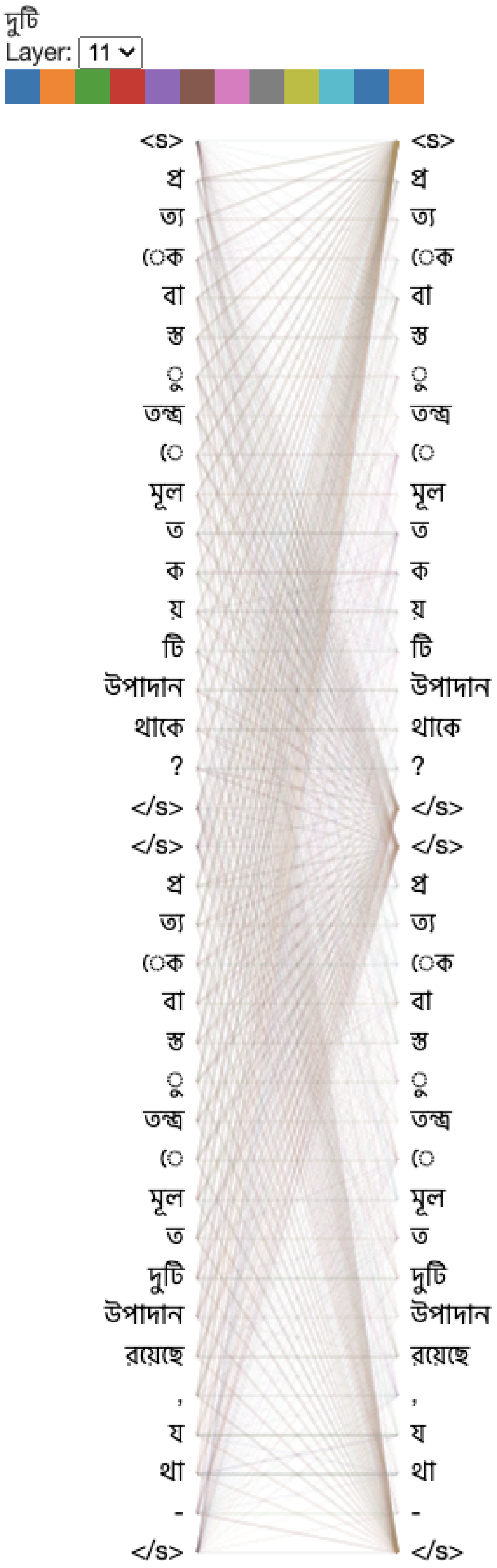

Figure 9: Binary classification attention: Bengali, Layer 11. Answer displayed at top left.

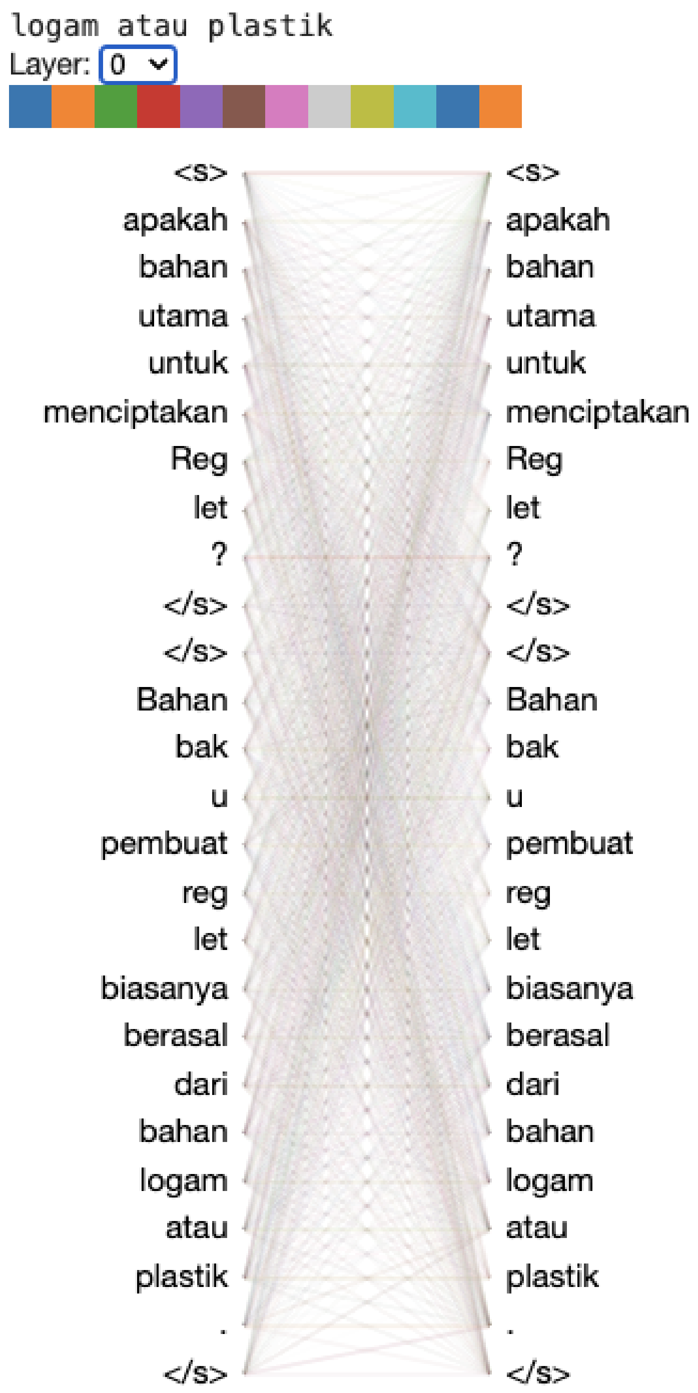

Figure 10: Binary classification attention: Indonesian, Layer 0. Answer displayed at top left.

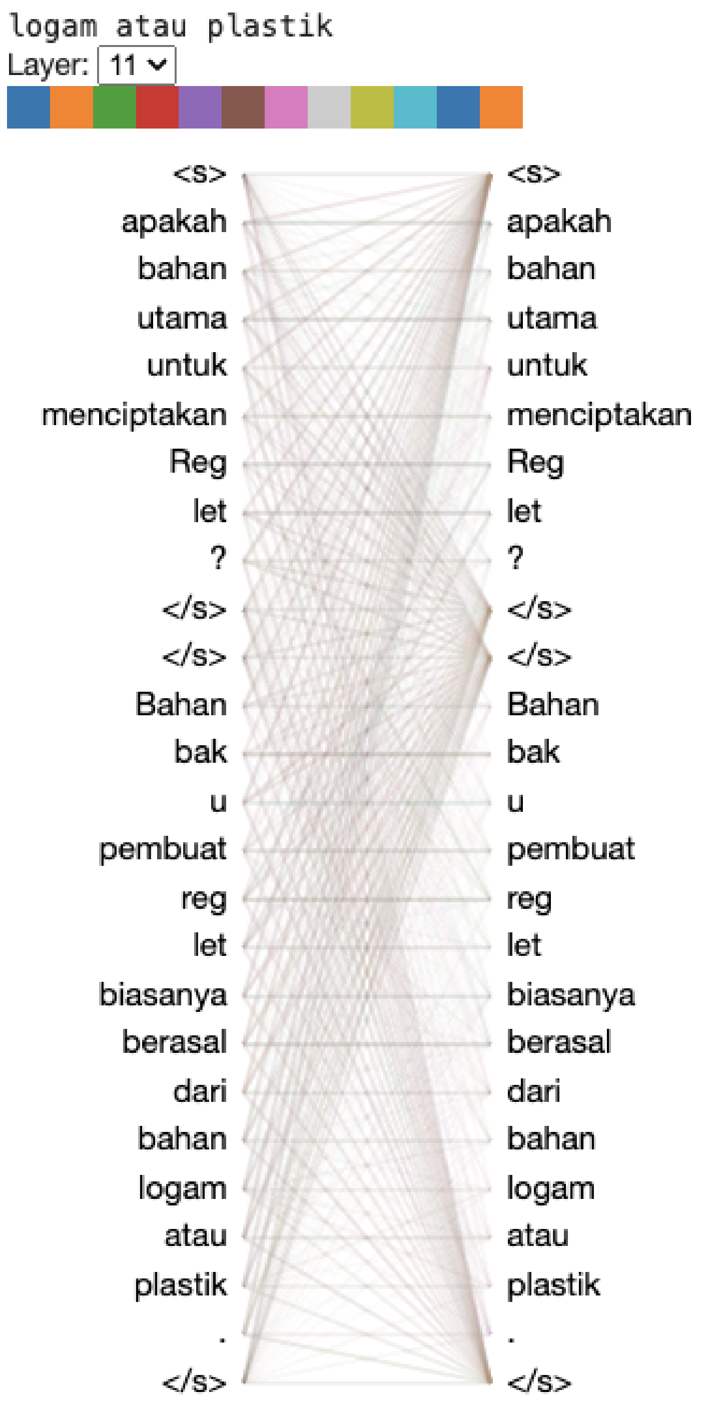

Figure 11: Binary classification attention: Indonesian, Layer 11. Answer displayed at top left.

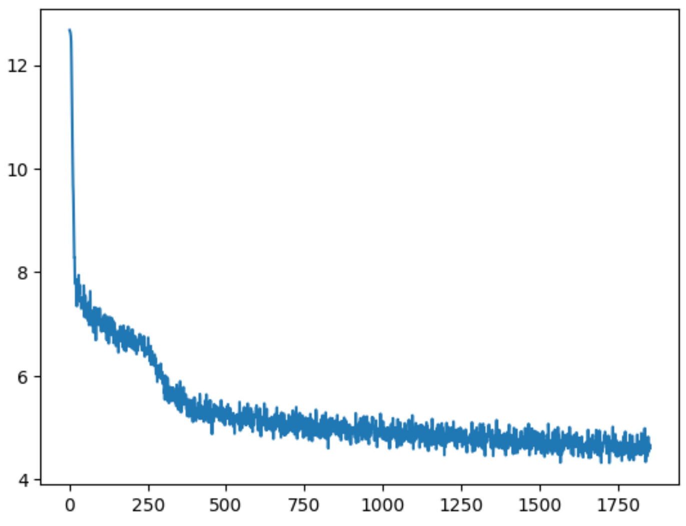

Figure 12: W37 LSTM: Arabic questions, loss

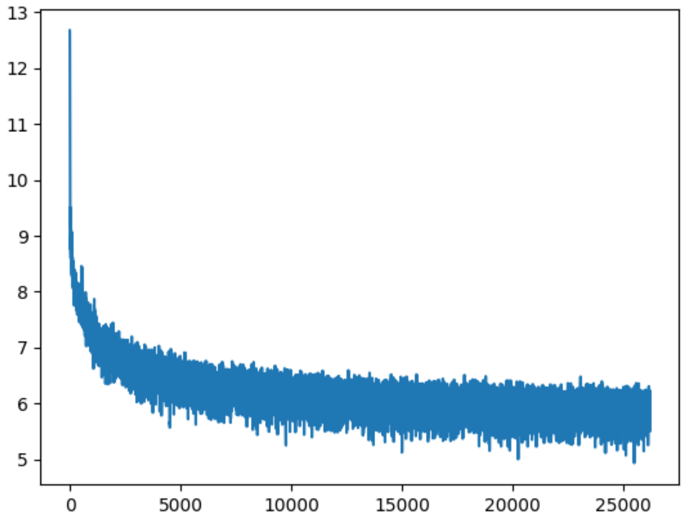

Figure 13: W37 LSTM: Arabic texts, loss

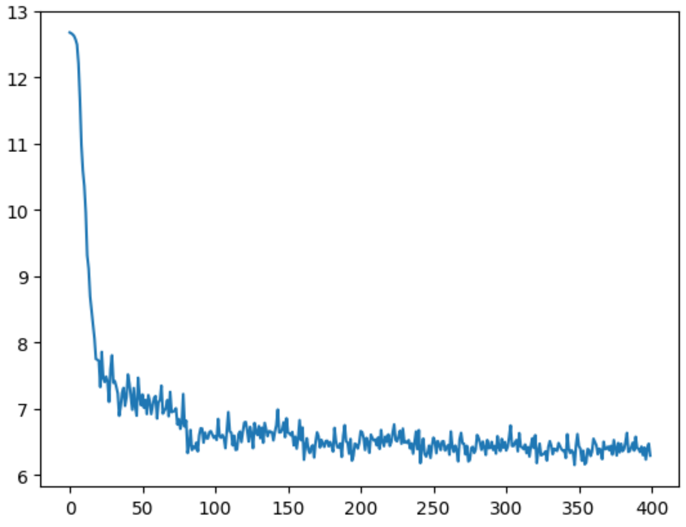

Figure 14: W37 LSTM: Bengali questions, loss

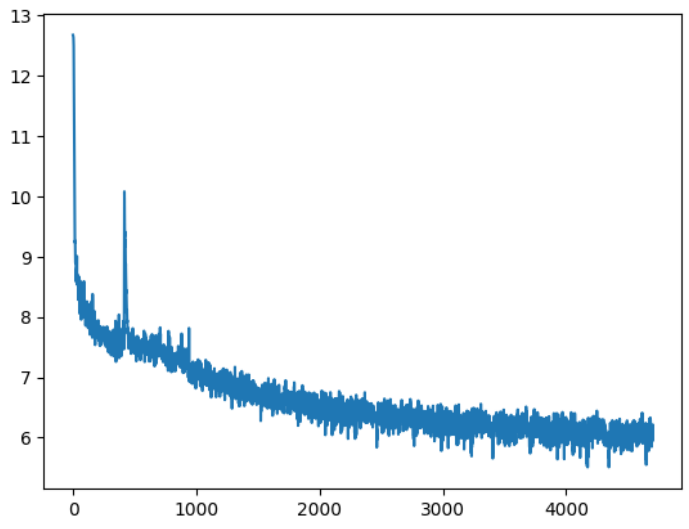

Figure 15: W37 LSTM: Bengali texts, loss

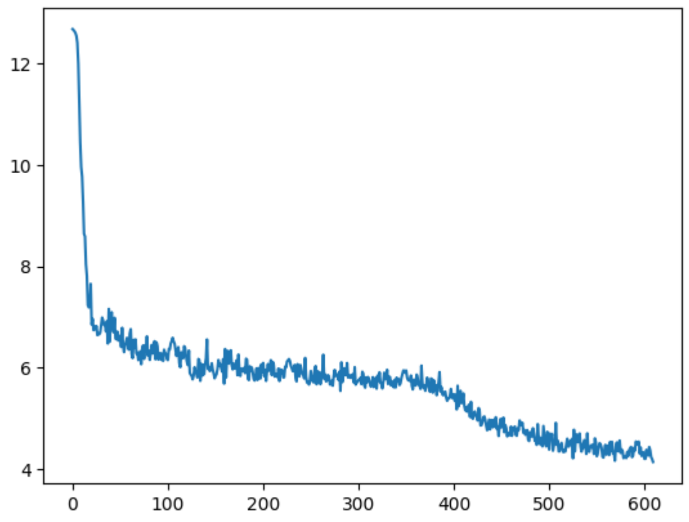

Figure 16: W37 LSTM: Indonesian questions, loss

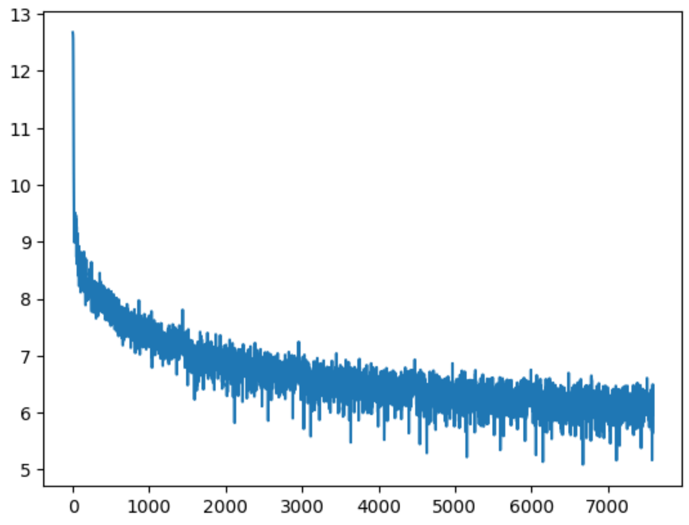

Figure 17: W37 LSTM: Indonesian texts, loss