Multimodal siamese neural network for similarity detection between (image, text) pairs

In this document, I will outline my approach to the deduplication task. The original dataset is multimodal, comprising two modalities: a tabular dataset containing title, cluster_id, link, and a counter identifier, and an image dataset stored in a folder. For the text component, I employed a sentence transformer, and for the image part, I used ResNet, both pretrained. The development process can be broken down into three steps.

Step 1: Preprocessing

In this step, my aim is to preprocess the datasets to make them ML-friendly and perform matching between the two modalities. Firstly, concerning the text modality, I observed that many titles were in languages other than English. To address this, I translated such titles into English. This was done because most sentence embedding models have been trained on English data and may not generate meaningful embeddings in a cross-language paradigm. However, after exploring the literature, I found that multilingual sentence embedding models have been proposed, and trying such an approach may be more suitable. There were also minor preprocessing steps, such as removing NaN values and discarding text rows that could not be matched with a corresponding image.

Consequently, I conducted matching between images and text, resulting in a uniform dataset with matched modalities. The matching is done once (i.e., for every sample in the original dataset I match a similar and a non similar sample). However, if resources were not an issue I would prefer to do an exhaustive match to result in a bigger and richer dataset.

The approach I then adopted is based on binary classification, where samples from the same cluster are considered similar, and two samples from different clusters are considered different. I believe that this can be naturally extended to the three-label case, where labels are similar, duplicate, and different.

Ultimately, the dataset was divided into training, testing, and validation sets.

Step 2: Model development

Following preprocessing, I obtained a multimodal dataset. A literature review revealed that Siamese models are an effective means of identifying similarities between items in a dataset. This neural network architecture employs the same weights while processing two different input vectors to generate comparable output vectors.

In light of this, I implemented a multimodal Siamese model, where each modality was processed by a Siamese model, and the outputs were fused. The overall approach is illustrated in Fig 1.

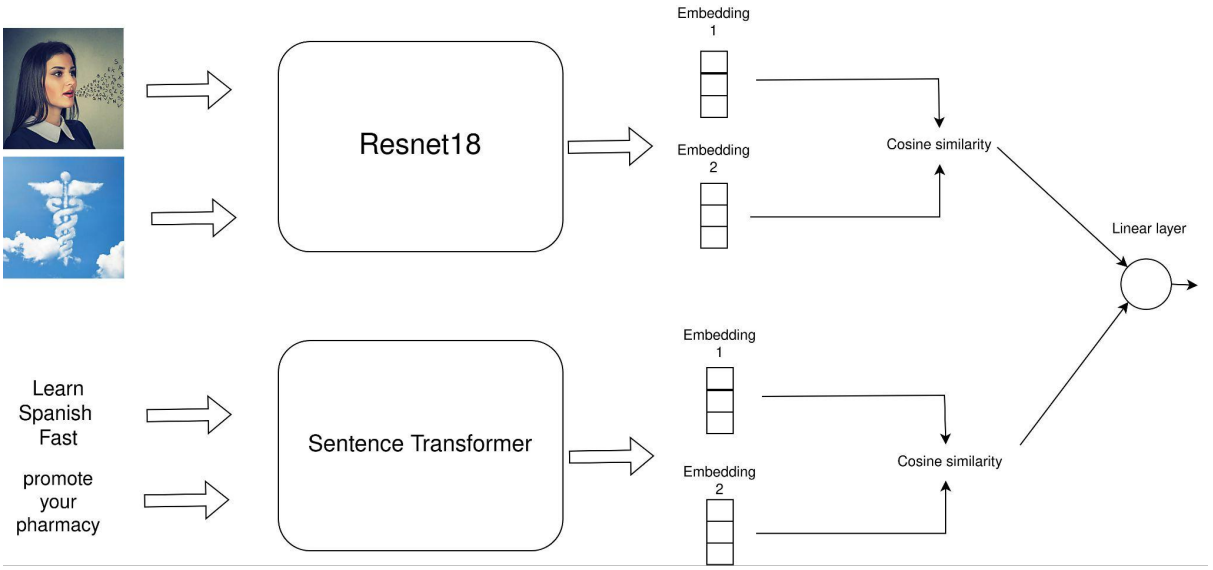

Figure 1: Siamese architecture designed for detecting similarity in (image, text) pairs through multimodal processing.

In this illustration, we observe that the (image, text) pairs undergo processing in their respective modalities, resulting in four embeddings—two for each modality. The two models, ResNet and Sentence Transformer, are already pretrained, providing a solid baseline. A simple averaging of the two cosine similarities yields satisfactory results. However, to tailor the model to our dataset, I introduced a single neuron that learns to weigh these two modalities through training.

Concerning the loss function, I utilized binary cross-entropy for the linear layer outputs, given the binary classification nature of the problem. Additionally, to ensure that the model produces embeddings that are farther apart in space for dissimilar pairs and closer for similar ones, I employed a cosine embedding loss using the intermediate embeddings produced by the Siamese models. The comprehensive loss is thus expressed as:

L = L(bce) + L(cosine_embedding1) + L(cosine_embedding2).

Arguably, this is not mandatory since the model's job is already to minimize the cosine loss indirectly. However, including such a loss introduces more direct enforcement during model training. I would like to test this with more training rounds and a bigger dataset. Moreover, training was conducted for 25 rounds (due to computational limitations), employing early stopping based on the validation loss. Results on the test set demonstrated approximately 93% accuracy and a 90% F1 score.

Step 3: API

Regarding the API to access this model, I have created four endpoints that serve in performing training, testing, and inference. As of now, I assume the user will clone the repository and enter the datasets manually; however, these could be downloaded or passed through an HTTP request (the data.txt file at least). There is also one endpoint that performs the preprocessing steps and creates a data.csv file, which can then be fed into the network. I added this endpoint because the preprocessing procedure is time-consuming (especially the translation through the googletrans API), and one needs to do it once usually. After that, through an API call, someone can train the model, which is automatically saved in the repository. Also, someone can test the model and get back metrics (accuracy, F1 score, etc.) through the test endpoint. To run the test endpoint, someone must have at least run the train endpoint once to produce a valid model. Additionally, I added an inference endpoint where someone can pass two images and two titles through an HTTP request, and the model will output a similarity score.