Database: The Guthenberg Project (20 books from Children's Literature)

Started coding on a Jupyter Notebook to clean the dataset, split the books into paragraphs and sentences and play around with embeddings

First embedding : Sentence-BERT (cosine similarity = semantic distance)

Next step : Find the mathematical formulation for finding an ordering that minimizes the distance between neighbouring sentences (order 1) and implement it, consider higher orders ?

Implemented and tested an algorithm that finds a local minimum for the minimal distance ordering problem with circular list of sentences

To test performance, but also if we want to eventually try a full end-to-end transformer network with pages as tokens, we need a loss function for orderings ⇒ distance over permutations :

$$\#swaps\left(\sigma_2^{-1}(\sigma_1)\right)$$

Researched on the structure of neural networks and understood the structure of the classifier to implement : a (freeze-)BERT + concatenation to introduce an asymmetry between A and B + linear fully connected + ReLU + linear n to 1 + sigmoid

Next steps : Alternative increment algo for minimal distance ordering, find an children's books author who has some 10-20 books in The Gutenberg Project, better distance on permutations, implement the classifier

Distance measures over permutations : Article p373, swap distance looks fine ?

$$\#swaps = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ \sigma_1(i)<\sigma_1(j)}}^{n} \mathbb{1}_{\sigma_2(i)>\sigma_2(j)}$$

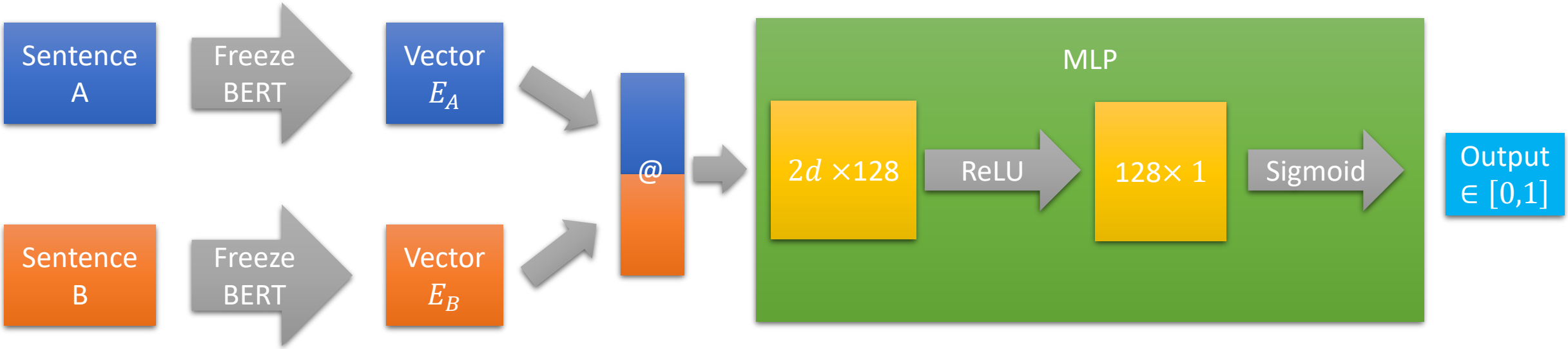
Implemented and sanity check -> the minimal ordering algorithm improves the swap distance significantly for a random permutation

As expected, the incremental algorithm for minimal ordering performs better on a solution already good, but way worse on a random permutation

Database of similar books : 27 books from the “Tom Swift” series by Victor Appleton and added an option to swap back and forth between databases

Next steps : Implement the full end-to-end transformer, complete default database, debug and finetune classifier (α), try to batch sentences for BERT

Classifier architecture



Other distance on permutations to compare – R-distance (unidirectional adjacency distance):

$$\#\{(i,j) \in \mathbb{N}^2, \quad \exists n_1, n_2, \quad (\sigma_1(n_1), \sigma_1(n_1+1)) = (\sigma_1(n_2), \sigma_2(n_2+1)) = (i,j)\}$$

Tutorial on PyTorch + rewriting the classifier with PyTorch

Should I have an activation function after the last layer ?

What mathematical formula to find a global ordering based on pairwise page orders ?

Tutorial on using VSCode as an IDE and setting up a coding project

Next steps : Implement the full end-to-end transformer, complete default database, finetune, test and **exploit** classifier (α/overfit), try to batch sentences for BERT, compact code

Corrected train/validation/test split on sentences and not pairs of sentences

Compacted code into several .py files imported in the Jupyter Notebook

Added the F1 score and the AUC for ROC curve → training on different books to generalize

If we embed the sentences of each book separately, we get different sizes of vectors → On what depends the output dimensions of BERT ?? → solution

Classifier validation for hyperparameter finetuning on different books,

$$epochs = 10\,000, \qquad \alpha = 0.01, \qquad L2 = 0$$

Created the greedy ordering algo in $O(n^2)$ which finds local minimum global ordering of sentences based on pairwise orders, permutation scores

$$\delta_{swaps} = ??, \qquad \delta_R = ??$$

Next steps : Implement the full end-to-end transformer, complete default database, try to batch sentences for BERT