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 ?

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

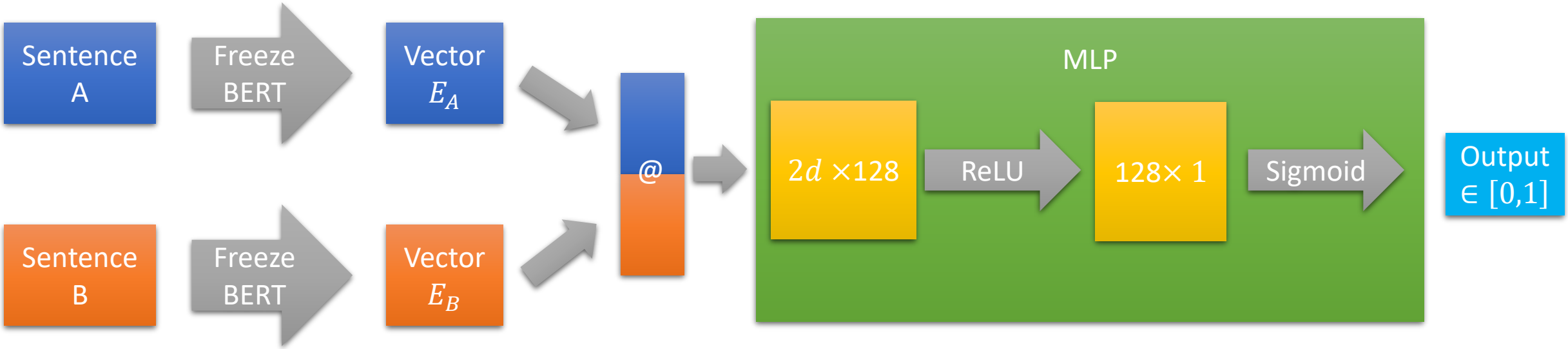
Implemented and sanity check -> the minimal ordering algorithm improves the swap distance significantly for a random permutation

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,

$$epochs = 1000, \qquad \alpha = 0.01, \qquad L2 = 0$$

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

We don’t get different sizes of vectors for different books, just a different number of page (fixed code)

Training on several books, test accuracy $= 58\%$, accuracy on new book $= 57\%$

Setup :
- EPFL VPN
- SSH to GPU server on VS Code
- GitHub Repository
- Running the code on the SSH

Parallelized computing “accelerate”

Report :
- Motivating the subject,
- Previous research on this topic, (https://www.jair.org/index.php/jair/article/view/12839/26707)
- Formal description of the problem and metrics,
- Why is this problem challenging and not yet solved,
- Different paradigms and formal description of the current classifier approach

Next steps : from BERT to JINA, batch sentences, database with pages, literature review for pairwise order to global order, implement the full end-to-end transformer, chapter clustering

Learned a bit about transformers, tokens, veq2seq, seq2vec, seq2seq, the underlying architecture of transformers and their advantages compared to LSTM, positional embedding, the multi-head attention mechanism, residual connections (equivalent of skip blocks in computer vision)

Literature survey on set-to-sequence problems in ML (40 pages) :
https://www.jair.org/index.php/jair/article/view/12839/26707

Design of the transformer end-to-end approach, main challenges are
- To design the output representing a distribution in the permutation space in a computationally reasonable way
- Design the network so it is not dependent on the input order – *ie* input set coding is permutation invariant or equivariant
- Alternatively, we can use a transformer with whole pages as tokens, but the transformer might not grasp any semantic meaning

There are hierarchical auto-encoder solutions (example) but we will focus on designing a solution using a permutation invariant input and attention pointer output based on Set Transformer

Next steps : from BERT to JINA, batch sentences, database with pages, implement the full end-to-end permutation invariant transformer, chapter clustering

Batching sentences for classifier, performance for 25 Tom Swift books 100% used :
80% accuracy and 80% $F1$-score for the test set
70% accuracy and 70% $F1$-score for a new book

Created a database from the current books with pagelike structure: The average length of a page in books is around 250 words which roughly corresponds to the limit of 256 tokens of the sentence-BERT we use

BERT to JINA: No need for the 8000 token length of JINA because the 256 token limitation of sentence-BERT isn't limiting us when processing 1 page, we switched sentence-BERT models from `all-MiniLM-L6-v2` to `all-mpnet-base-v2` that has max token size of 384 instead

Report:
- formal description of the current classifier approach
- 2 challenges of the set-to-sequence problem
- permutation invariance is not necessary and equivariance is enough
- description of the Set Transformer approach

Next steps : implement the full end-to-end permutation invariant transformer, chapter clustering, maybe TSP genetic algorithm

TSP: Adding the transitivity hypothesis (page1 before page2 and page 2 before page 3 implies page1 before page3)
- Enforce it to make the results more consistent (min weighted transitive closure with Floyd-Warshall)

$$w_{i,j} = \frac{1}{\left|\text{logit}\left(p_{i,j}\right)\right|}$$

- Use it to get the permutation with topological sort

$$\delta_{swap} = 0.125, \qquad \delta_R = 0.897$$

For comparison for a random permutation of length 156,

$$\delta_{swap} = 0.5, \qquad \delta_R = 0.994$$

Report:
- Transitive closure
- TSP approximation analysis
- token reordering vs embedding reordering

Next steps : implement permutation invariant method, maybe TSP genetic algorithm, TSP polynomial 1.5-approximation (Karlin), chapter clustering and intent detection (COPA)

Implementing the set transformer

DRAWING

Test

Improvements on the set transformer (add to report)

Adaptions from the set transformer (add to report)

Old English tokenizer

Next steps : improve permutation invariant method, maybe TSP genetic algorithm, TSP polynomial 1.5-approximation (Karlin), chapter clustering and intent detection (COPA)