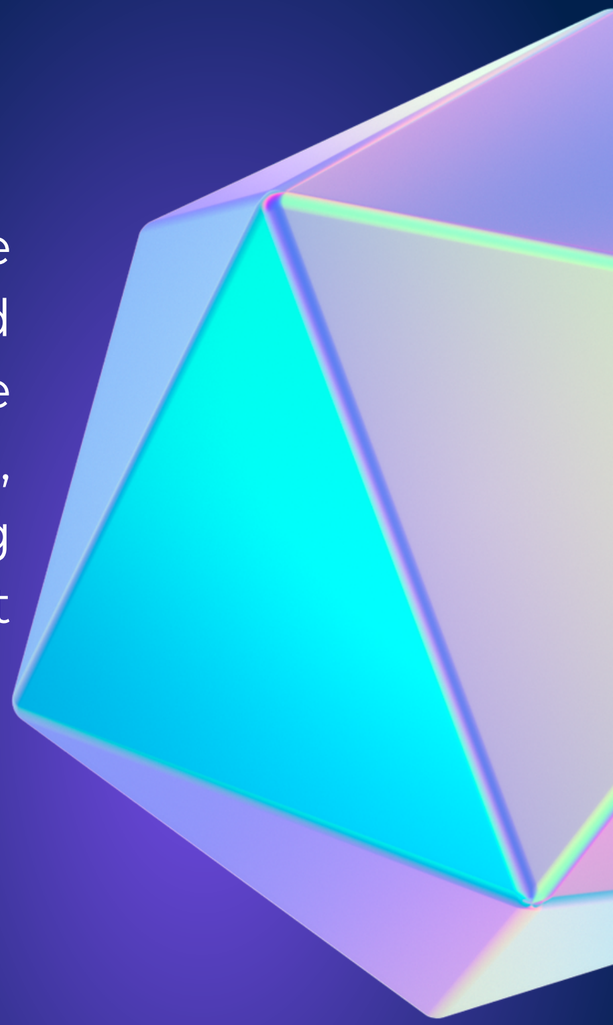
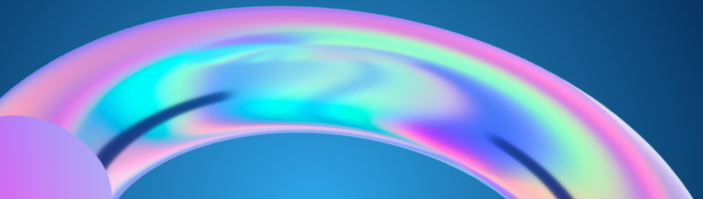
Team-29

DevRev Problem Statement

Abstract



Our final pipeline is a three-step approach to solve the retrieval and question-answering task. We majorly used transformers to form the embeddings and extract the answers. The pipeline consists of the Retriever Module, Re-Ranker Module and the Reader Module. These along with some clever optimisations helped us achieve great scores.



Literature Review

- Unsupervised Document Embedding via Contrastive Augmentation by Luo et al. [1]
- Adaptive Margin Diversity Regularizer for handling Data Imbalance in Zero-Shot SBIR by Dutta et al. [2]
- Towards Universal Paraphrastic Sentence Embeddings by Wieting et al. [3]
- SimCSE: Simple Contrastive Learning of Sentence Embeddings by Gao et al. [4]

Implementation Roadmap

Paragraph Retrieval

- Context/Query encoder
- Similarity Metric
- Experiment with Retriever:

A)Classification with Adaptive Margin Regularisation
B) Contrastive Learning

Re-Ranking Module

- New and different cross encoder models

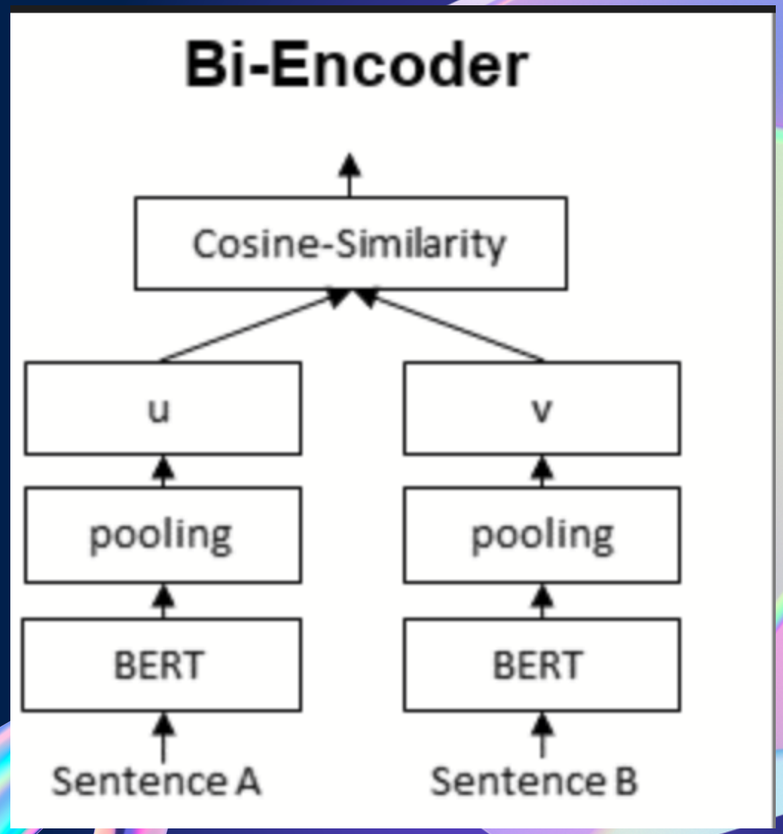
Reader Module

- Extraction of best answer text from retrieved passages

Paragraph Retrieval

Context/Query Encoder:

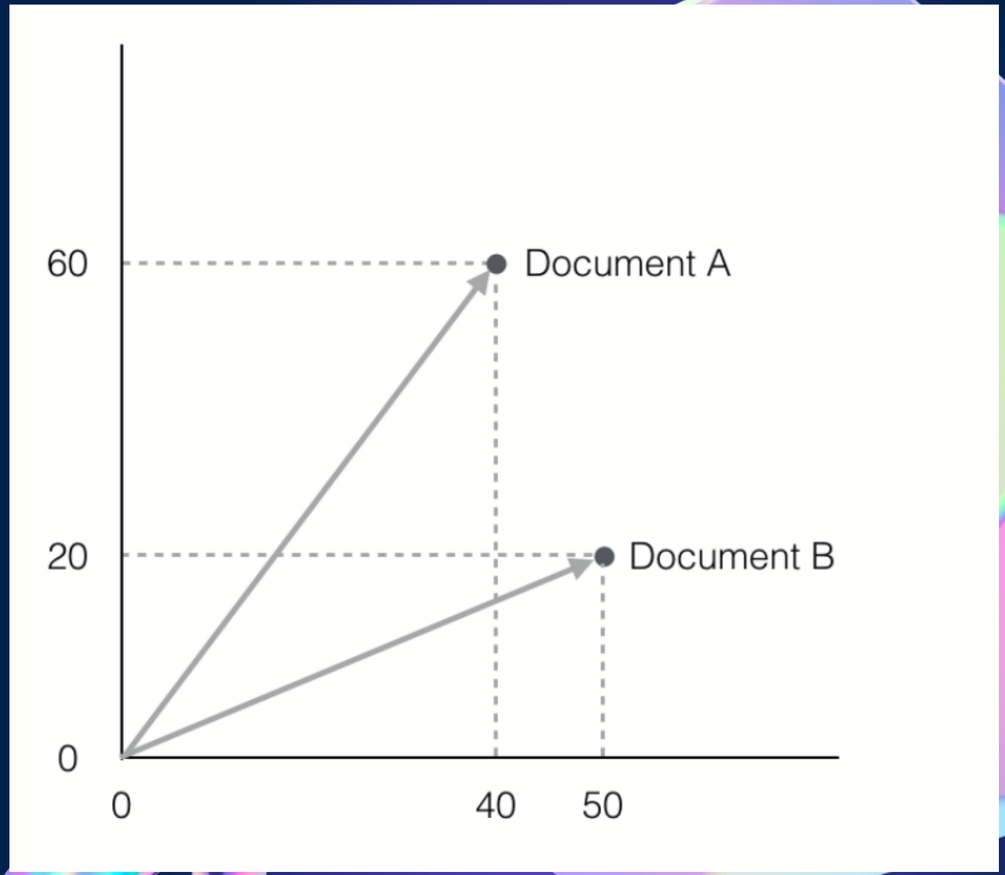
- We experimented with various models and methods for encoding, like
  - Word2Vec Averaging,
  - FastText,
  - Wiki2Vec
  - Doc2Vec
- Transformer-based models, including
  - Sentence BERT,
  - Google Universal Sentence Embeddings
  - a ROBERTA-based model
  -



Paragraph Retrieval

Similarity Metric

- We used cosine similarity as our similarity metric and found that it performed better than the slower euclidean distance.
- The query and context encoder representations were stored in a vector database
- We selected the top-K most similar contexts for each query and passed them to the re-ranking model.



Research on Query encoder models

Google Universal Sentence Encoder

FastText

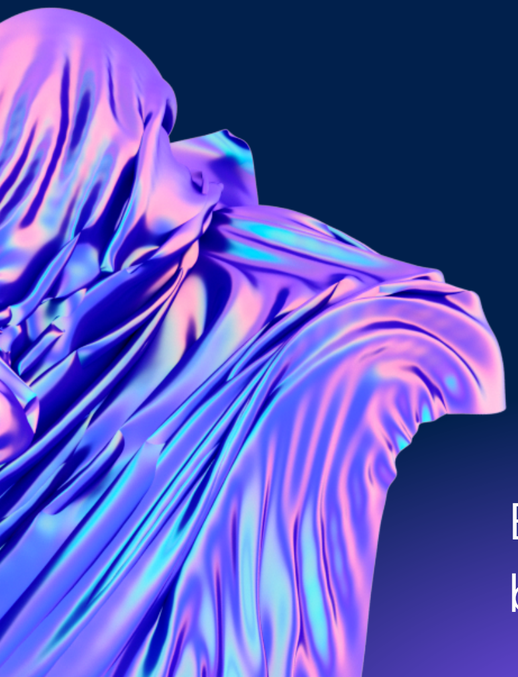
Sentence Bert

Word2Vec

Doc2Vec

Method Name	Latency (in secs)	Accuracy
Word2Vec Averaging	0.002	38%
FastText	0.002	44%
Doc2Vec	0.0025	52%
Google Universal Sentence Encoder	**0.2**	**74%**
Sentence BERT	0.35	60%

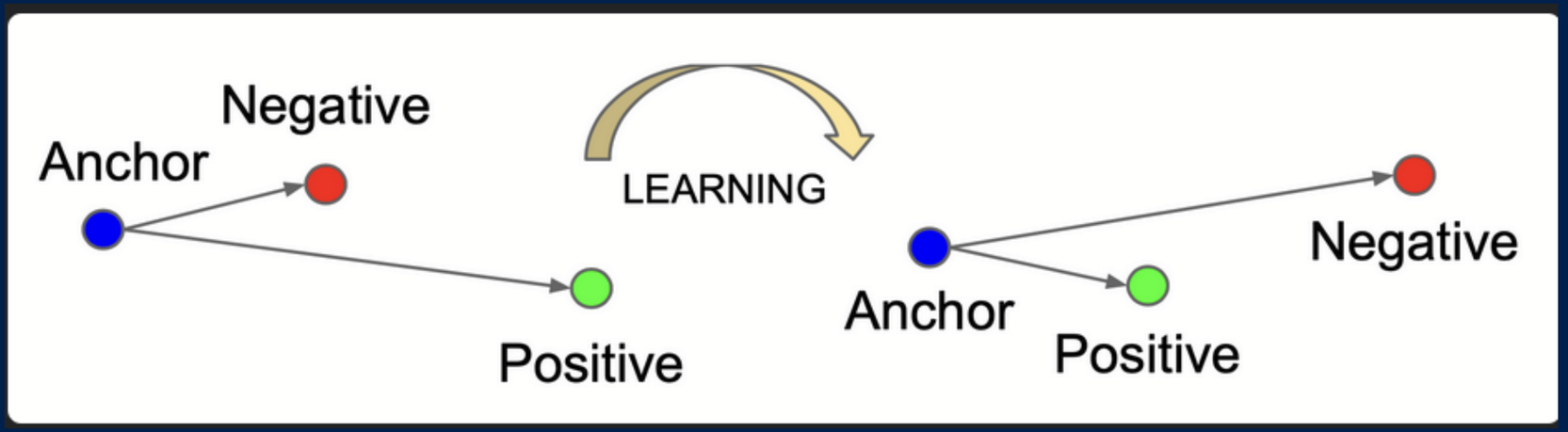
Experiments With Retriever



We also conducted several experiments to improve the embeddings generated by the encoder. The two main approaches were:

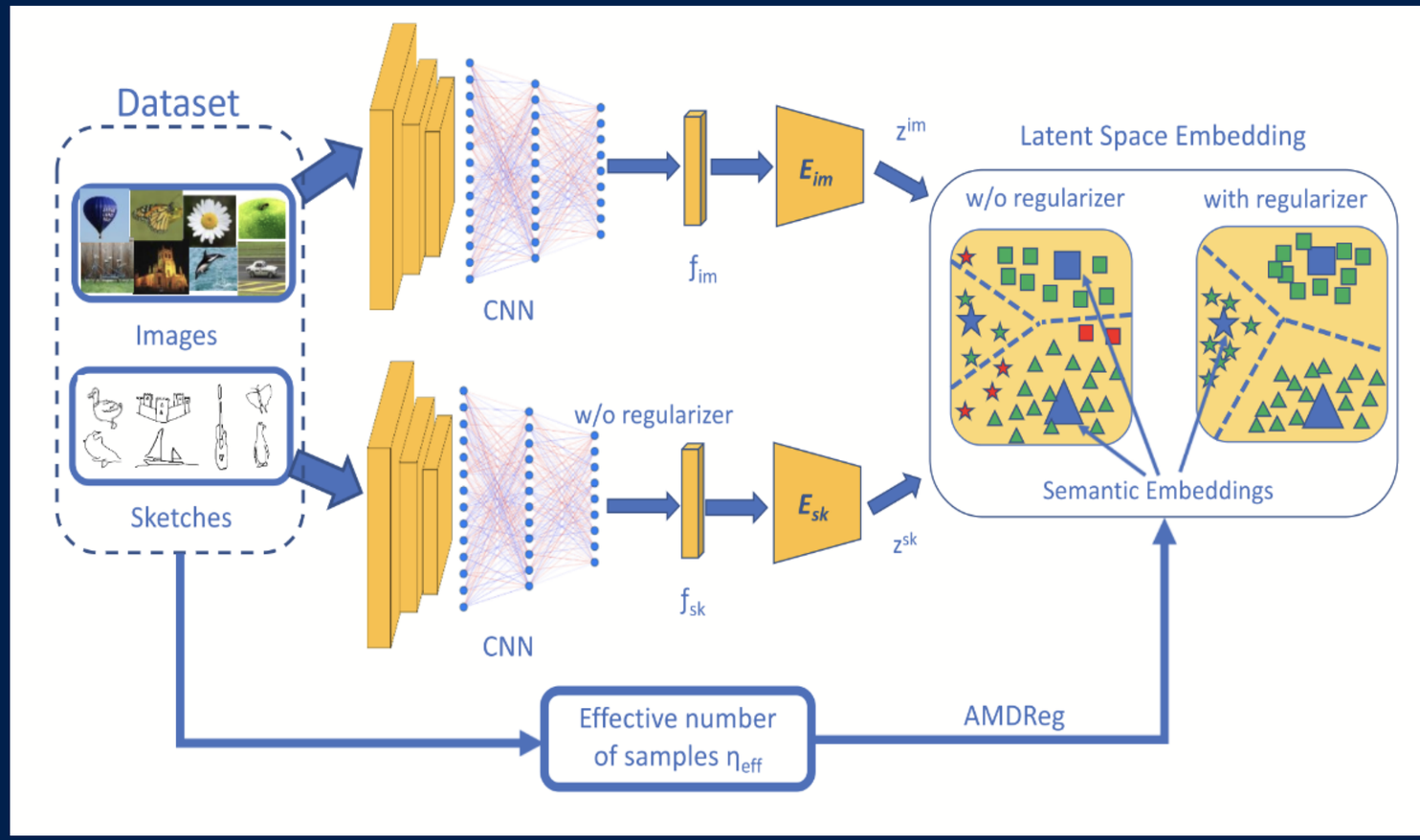
- Contrastive Learning:
  - Cluster embeddings by fixing context embeddings as anchor points
  - Contrastive Loss brings relevant question embeddings closer in the latent space.
- Classification with Adaptive Margin Regularization:
  - Bring the context embeddings and query embeddings in the same latent space
  - Added a loss which increases the margin between two context points.

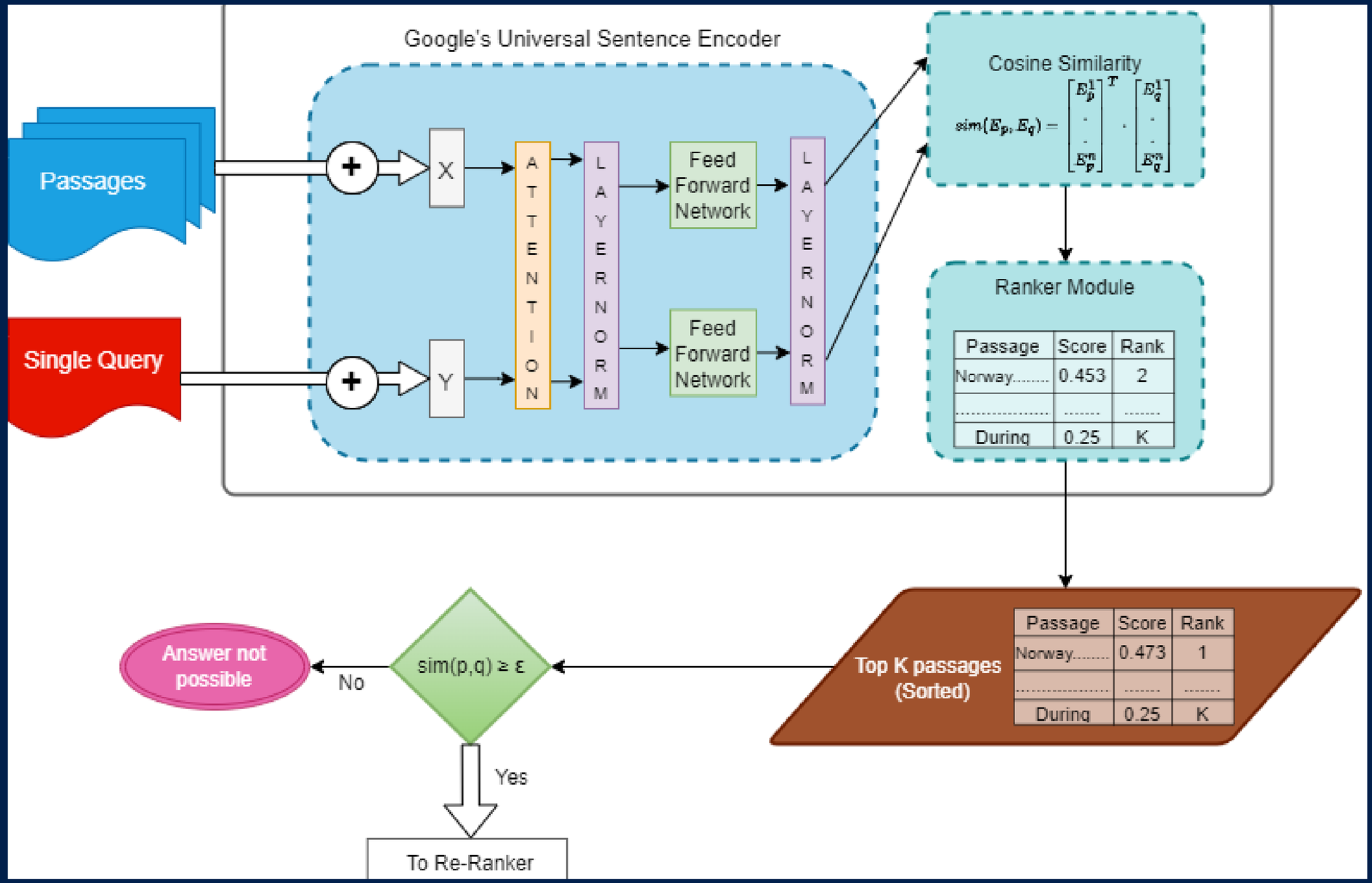
Both approaches have potential for improving retriever accuracy and would benefit from more data and time.



Visualisation of the Contrastive Loss

Diagram of how the AMDReg Works. (Diagram taken from the cited paper [2])

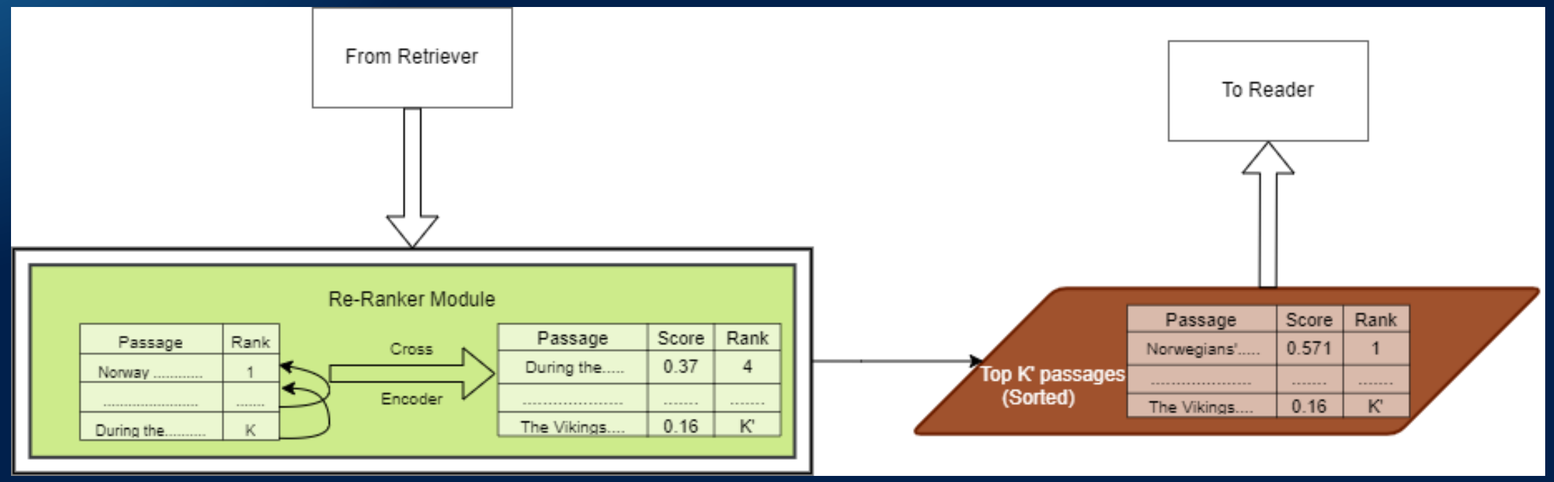




Retrieval Pipeline

Re-Ranking Module

To improve the performance of the vanilla retriever model, we introduced a cross-encoder model that re-ranks retrieved contexts and questions based on their answerability. This significantly improved the performance metric. We tried three cross encoders (MiniLM, Roberta Tiny, and Tiny BERT) and Tiny BERT proved to be the fastest and most efficient, given the hardware constraints.

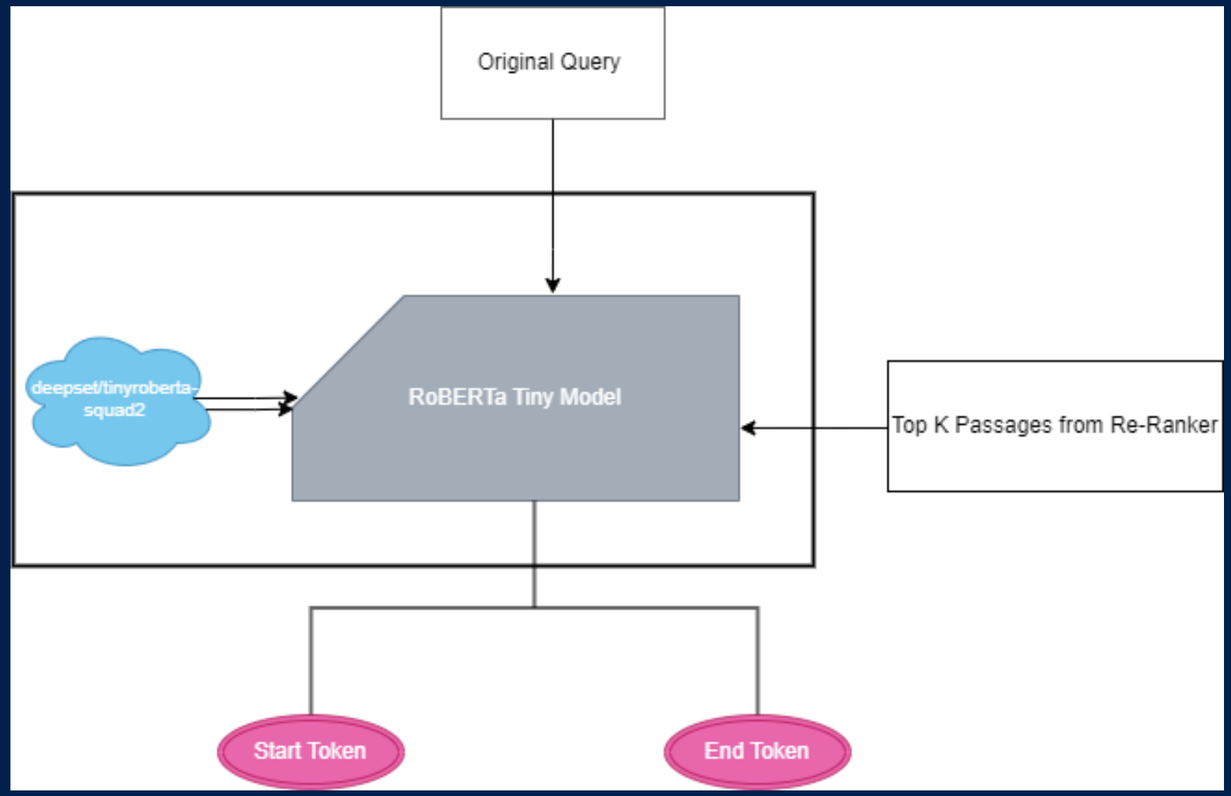


Re-Ranker Module

	Evaluation Metric
Score Without Re-Ranker	0.65
Score with Ranker	0.74

Reader Module

The final step in the pipeline was to use a model for extracting answers from the retrieved and re-ranked passages. We used the same model from our mid-evaluation as it had the best results, being both fast and having the best metric.



Reader Module

Optimisations

Choice of Language Models Used

- Light weight models like MiniLM, Tiny BERT, and RoBERTa Tiny
- The best model was selected by evaluating them on a 25% split of the training data.

Pre-Computing The Context Embeddings

- The context embeddings were pre-computed and stored in a vector database.
- Only new question embeddings were calculated during inference.

Parallelising on the CPU

- The use of Ray Library increased performance by allowing parallel processing of the model on multiple CPU cores
- Google Colab's free tier offers 2 CPU cores, which only reduced our inference time by half.
- Our inference script can scale to multiple CPU cores for even faster processing.

Other Experiments and Ablation Studies

Generative Pseudo Labelling and Data Augmentation:

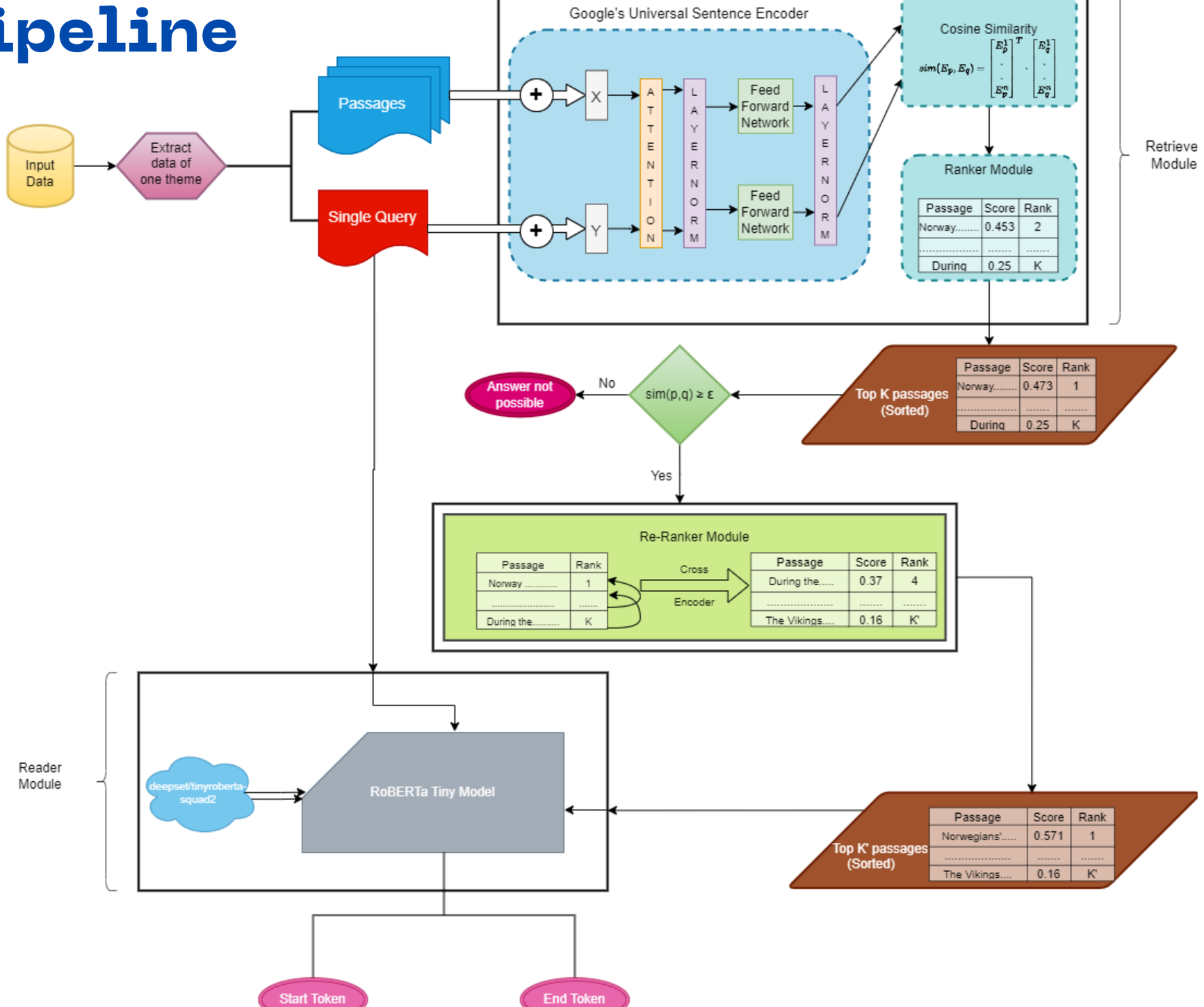
- We tried two methods to address the issue of unanswerable questions and insufficient questions in some themes
  - Pseudo Question Creation.This took too long **(1.5 hours for 10 questions)** and was not feasible for the entire dataset.
  - Data augmentation techniques like sentence reordering, synonym replacement, random swaps, and random deletions.

Making Theme Specific Models :

- Fine-tuning the RoBERTa Tiny reader model on each theme was tried to make it specialized in answering questions from a particular theme,
- However, this resulted in overfitting to the training data and a drop in score to **0.53**. The generalization of pre-trained models on the SQuAD v2 dataset proved to be more effective.

Method	Train Time	Final Score
Pseudo Label Generation	9 minutes per question	N/A
Data Augmentation	0.462 seconds	0.63
Specific Reader Models	20 mins for 1 epoch	0.53

Final Pipeline



Final Result and Conclusions

Our pipeline achieved a 0.74 score for paragraph retrieval and 0.71 for question-answering, an improvement over the mid-eval results. Lack of data and inability to use external data hindered further improvements through better hyperparameter tuning.

Future Work

There are a few possible ways to further improve the quality of the retrieved documents. We list a few here:

- Averaged Query Expansion:
  - Use the top-k retrieved document and average their embeddings
  - Forms a query from within the distribution of the contexts.
  - Re-retrieve the top-k passages using this query
- Increase per context query size.
- Work on improving document embeddings i.e. use the increased dataset to train models which use methods like AMDReg

Questions?

Thank You