Recognizing Textual Entailment on Customer query and reason for the query

Name:- Hritik Rajendra Akolkar
Email Id:- hritikakolkar@gmail.com
Linkedin :- https://www.linkedin.com/in/hritikakolkar

Problem Statement

Given a text and a reason, predict if **text** satisfies the **reason**. You can use the `train` file for any training and report metrics on `evaluation` file.

Note: Small train dataset with only positive samples is intentional. You should generate negatives on your own. You cannot use eval samples for training.

Some analysis of the given dataset

The dataset is based on the problems, query and feedback user have, when they use a certain product and we have the reason for every query.
After randomly sampling 100 data points from training dataset I found that the data is correctly labelled but problem lies in evaluation dataset after looking at some data point found that some negative samples are incorrectly labelled.
The Training file consists of only 2061 positive labels training examples and Evaluation file consists of 3001 positive labels and 5999 negative labels with total of 9000 evaluation examples. We are using metrics like precision, recall and f1-score which are robust to Imbalanced dataset.
Minimum No of words in training text:- 4
Maximum No of words in training text:- 66
Average No of words in training text:- 11.40

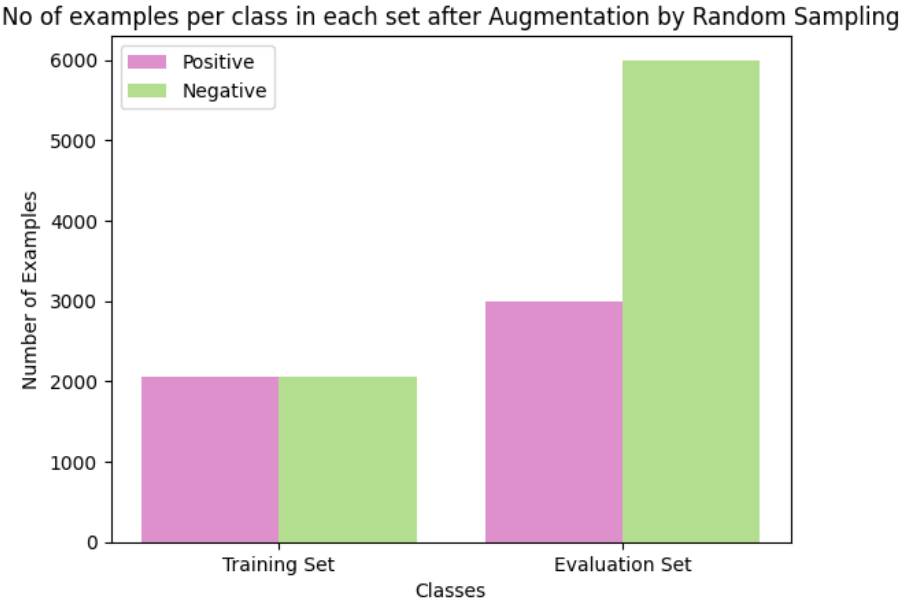
Minimum No of words in training reason:- 2
Maximum No of words in training reason:- 16
Average No of words in training reason:- 5.48

Data Processing and Augmentation

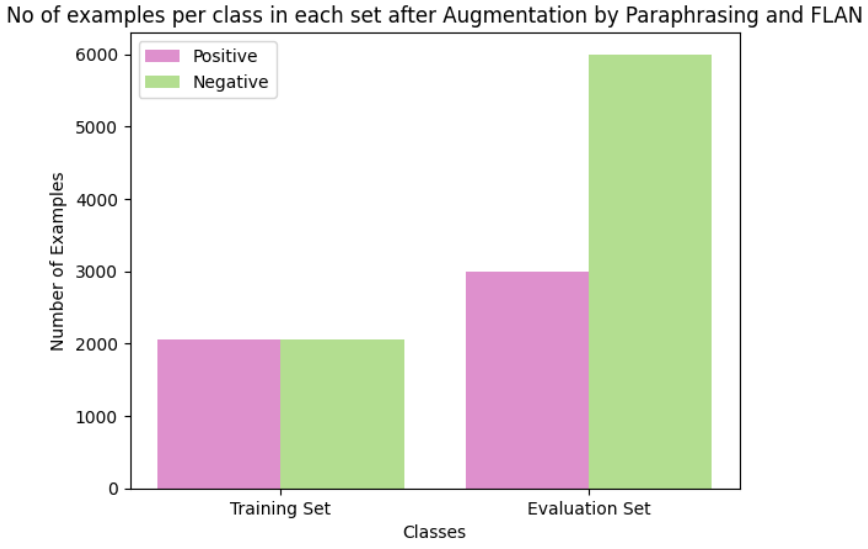
Data Cleaning :- The dataset is almost clean so no need of cleaning. Still cleaned some external brackets.

Data Augmentation :- There are multiple text augmentation concepts which can help to increase the diversity of the training data and improve the generalization ability of the model. Some commonly used NLP augmentation techniques include character level, word level augmentation like (synonym replacement and using word2vec embeddings to replace with most similar words) , Context-based augmentation like (back translation and Paraphrasing). Back translation involves translating the sentence pairs from one language to another and then translating them back to the original language. Paraphrasing is expressing the same message as the original text using different words or phrases without loosing the meaning of the message.

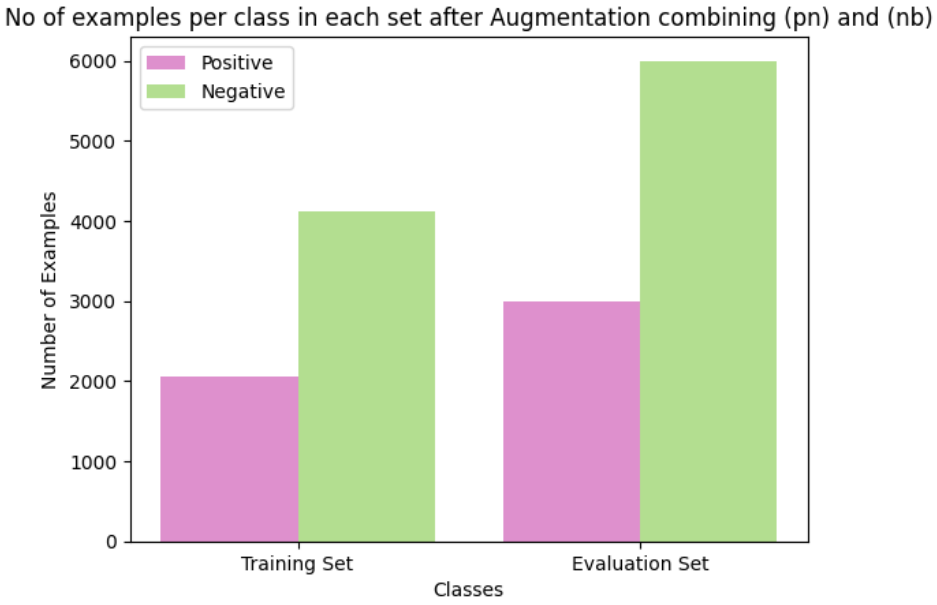
1. Random Sampling: (pn)



2. Use Parrot Paraphraser for paraphrasing only the query and combined it with the negatively generated reason by flan-t5-base: (nb)



3. Combined both Random Sampling (pn) and (nb):



Model Selection and Architecture

There are two ways to solve textual entailment problem one is using sentence Transformers for the purpose and second is based on Language Models with Instruction Finetuning (FLAN). For selecting the right architecture for the model I am experimenting with bi-encoders with loss function like Contrastive Loss, MNR loss, and cross-encoder architecture. I also experimented with finetuned and hyperparamenter tuning of FLAN models (Question Answering template) using **wandb** sweep.

Setting up Human Base Line Error:-

Randomly Sampled 100 examples from 9000 evaluation data points. Then manually labelled according to he conditions and the problem statement. The problem with this is while labelling I was having doubts and was not clear about the conditions.

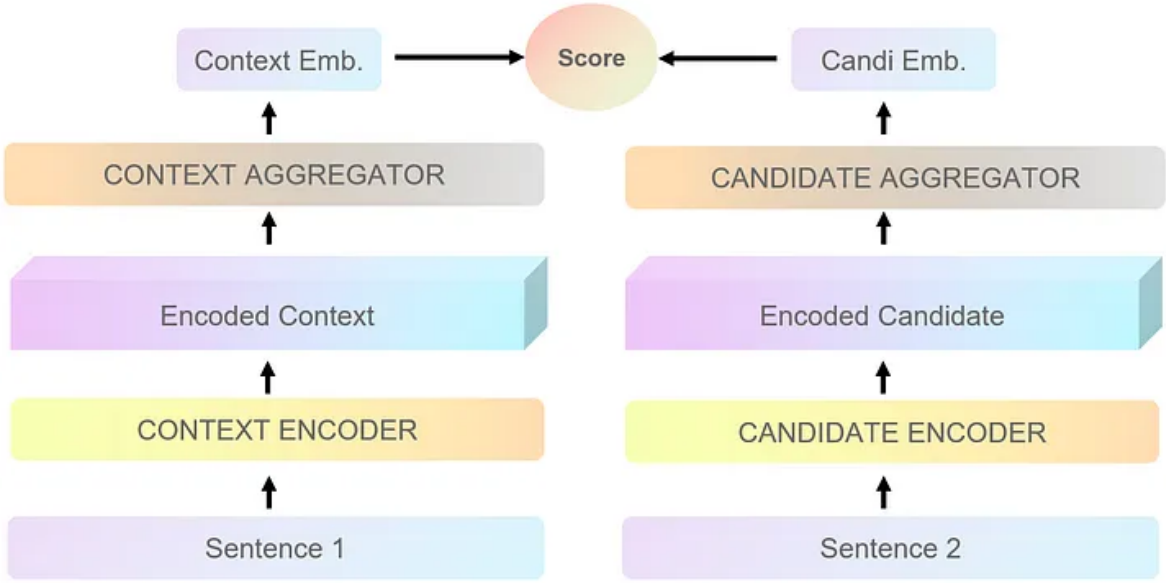
Metrics	Score	Error
Precision	0.54	46%
Recall	0.86	14%
F1-Score	0.66	34%

Model Architecture :-

Sentence Transformers

Using a Siamese architecture will be a good choice as we are comparing two sentences (query and reason) to predict whether the reason satisfies the query. This architecture uses two identical models to encode the two input sentences separately, and then a distance metric is applied to the encoded representations to predict the similarity or dissimilarity between the two sentences.

1. **Bi-Encoders :-** Bi-Encoders for a given sentence produces a sentence embedding. We pass to a Model independently the sentences A and B, which result in the sentence embeddings u and v. These sentence embedding can then be compared using cosine similarity. We can use multiple Loss Functions like Cosine Similarity Loss, Triplet Loss.



Credits : Chein Vu

Using pretrained model to finetune on the augmented datasets

- all-MiniLM-L12-v2: Sentence Transformer that maps sentences & paragraphs to a 384 dimensional dense vector space. The model uses pretrained microsoft/MiniLM-L12-H384-uncased model and fine-tuned on a 1B sentence pairs dataset. The use a contrastive learning objective: given a sentence from the pair, the model should predict which out of a set of randomly sampled other sentences, was actually paired with it in our dataset.

Approach: Used Contrastive Loss for finetuning the model (pn + nb) dataset

Contrastive Loss:-

Given a pair of data points, let x1 and x2 be the embeddings (or representations) of these data points. Let y be the label indicating whether these two data points are similar (y=1) or dissimilar (y=0). The contrastive loss can be defined as:

$$L(x_1, x_2, y) = 0.5((1-y) \cdot d(x_1, x_2)^2 + y \cdot \max(0, m - d(x_1, x_2))^2)$$

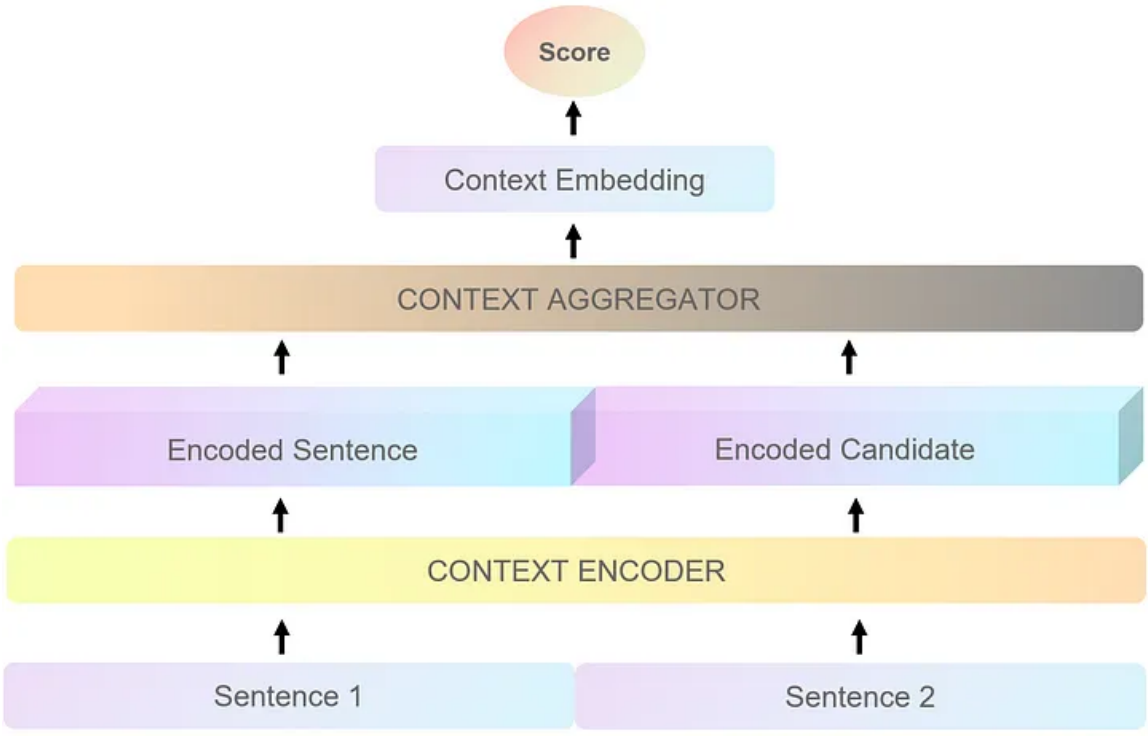
where d(x1, x2) is the distance between the two embeddings x1 and x2 (used Cosine Distance), and m is a margin parameter that controls the minimum distance between similar points. If the distance between the embeddings of similar points is less than m, then the loss is zero; otherwise, the loss increases as the distance between the embeddings increases.

The model showed good results on the (pn + nb) and generalized well. Similarly I experimented on the other bi-encoders sentence transformer architecture including Bert and Mp-Net Based to see their results and most of the large models using Contrastive Loss overfitted on the dataset as we can see in table 1.

Multiple Negative Ranking Loss:- The loss function encourages the model to rank the positive examples higher than the negative examples. This loss expects as input a batch consisting of sentence pairs (a_1, p_1), (a_2, p_2)…, (a_n, p_n) where we assume that (a_i, p_i) are a positive pair and (a_i, p_j) for i!=j a negative pair. For each a_i, it uses all other p_j as negative samples, i.e., for a_i, we have 1 positive example (p_i) and n-1 negative examples (p_j). It then minimizes the negative log-likehood for softmax normalized scores.
This loss function works great to train embeddings for retrieval setups where you have positive pairs (e.g. (query, relevant_doc)) as it will sample in each batch n-1 negative docs randomly.

Started with all_datasets_v3_mpnet-base using MNR Loss, this method showed significant improvement in the metrics trained on only positive set provided. Finally experimented with more layers of models like all_datasets_v3_roberta-large which gave f1-score of 0.68 (all time high).

2. **Cross-Encoders :-** Unlike bi-encoder models that produce vector embeddings for individual data instances, cross-encoder models take a pair of data items as input and use a classification mechanism to predict the relationship between the two items. Therefore, it is not possible to pass individual sentences to a cross-encoder model. Instead, the model requires a pair of sentences that need to be classified as entailed or non-entailed. This makes cross-encoder models particularly suitable for tasks such as textual entailment and paraphrase identification, where the goal is to compare and classify pairs of sentences.

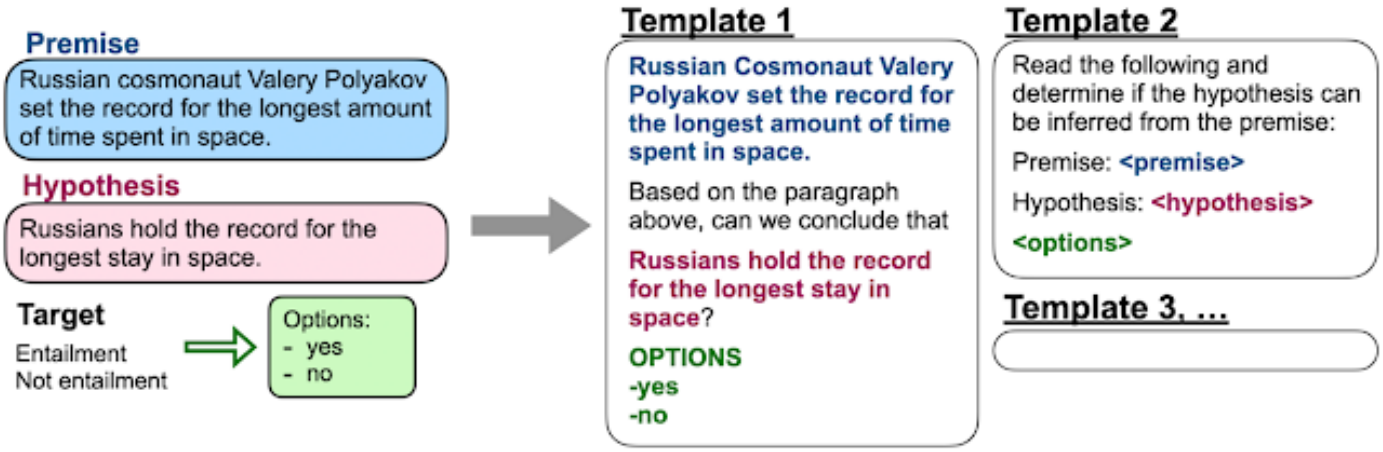


The finetuned Bert base model was experimented with using three augmented datasets, but the results were not satisfactory as the model was overfitting on the training set. To improve the cross-encoder model, hyperparameter tuning is necessary. However, as the bi-encoder model with MNR loss has shown promising results, it would be a good idea to focus on improving this model.

FLAN (Fine-tuned LAnguage Net)

FLAN fine-tunes a pre-trained model on a large set of varied instructions that are generated from existing datasets using templates. This approach allows the model to learn to follow instructions in a language-agnostic way, meaning it can understand and follow instructions in multiple languages without requiring specific training for each language. One of the key insights of the FLAN technique is that by training a model to follow a diverse set of instructions, it can learn to generalize to new instructions and tasks more effectively. This is because the model is forced to learn to extract useful information from the instructions, rather than simply memorizing a set of specific examples.

FLAN models, based on question-answering templates, are trained to generate natural language responses to given prompts or questions. These models are pre-trained on large datasets, and finetuning on a smaller dataset can improve their performance on specific tasks.



I have chosen to utilize the T5 transformer model for our Question Answering (QA) task, as it has achieved state-of-the-art performance on several QA benchmarks. While there are other models, such as GPT-3 and the BERT family, T5's multi-task learning approach enables it to learn from various tasks simultaneously, which can enhance its performance on specific tasks like QA.

The FLAN-T5 model is fine-tuned by providing input text in the form of a query and reason as "Q" format, with the output limited to "yes" or "no". For instance, the input_text may include a query like ***"Did the user query '{query}' satisfy or justify the reason '{reason}'?"*** and the corresponding label_text will be either ***"yes"*** or **"*no*"**.

For example, a real input_text could be ***"Q: Did the user query 'I am not able to download this app' satisfy or justify the reason 'want to download the app'?"*** with the corresponding label_text "***yes***". During the fine-tuning process, the maximum number of new tokens is limited to 2 and the length penalty is set to 4.0.

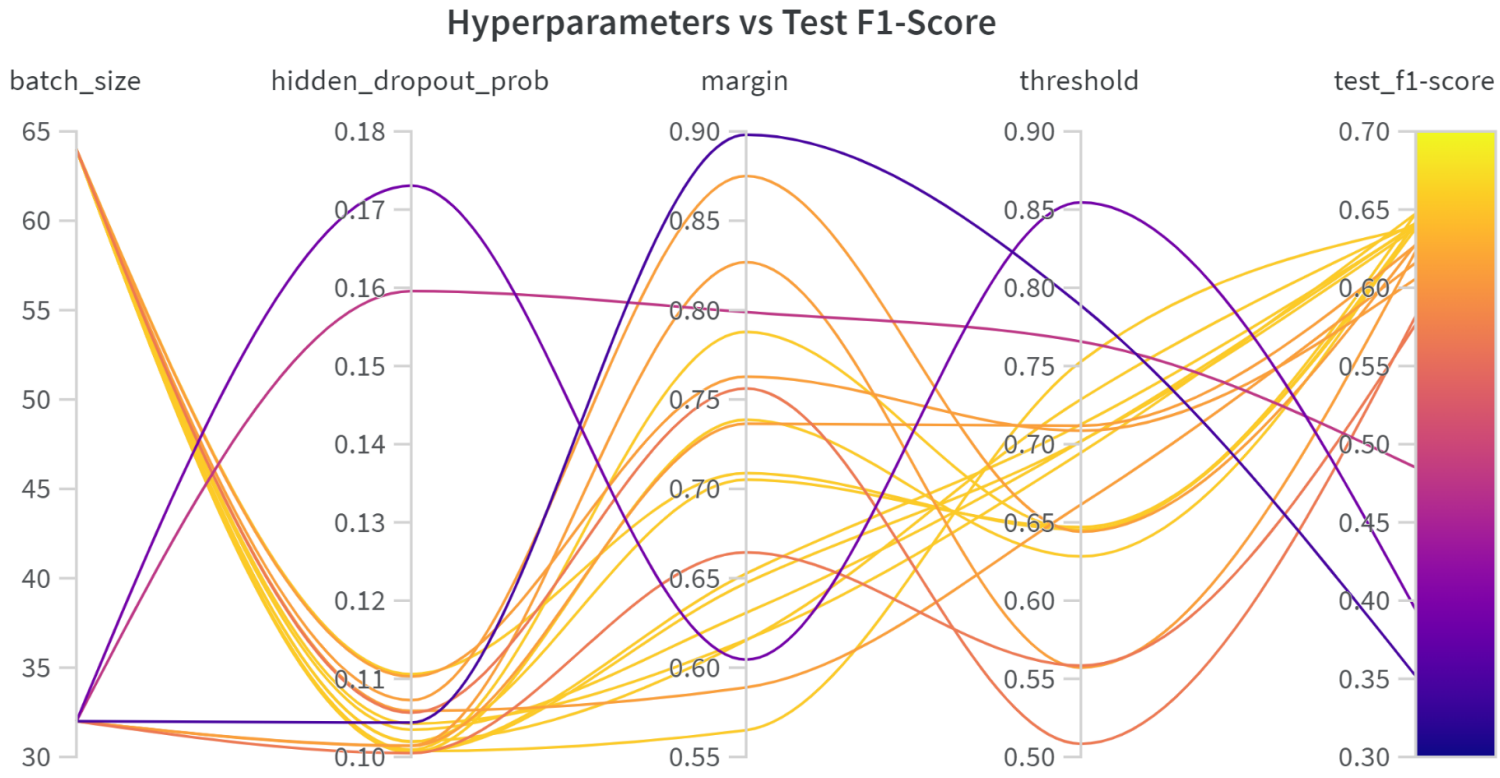
After experimenting with different versions of the FLAN-T5 models, we found that the FLAN-T5-Base model performs well and can be fine-tuned with our available computational resources.

Table 1. Ablation Study on the Evaluation data (* Baseline Approach , ** Selected Approach)

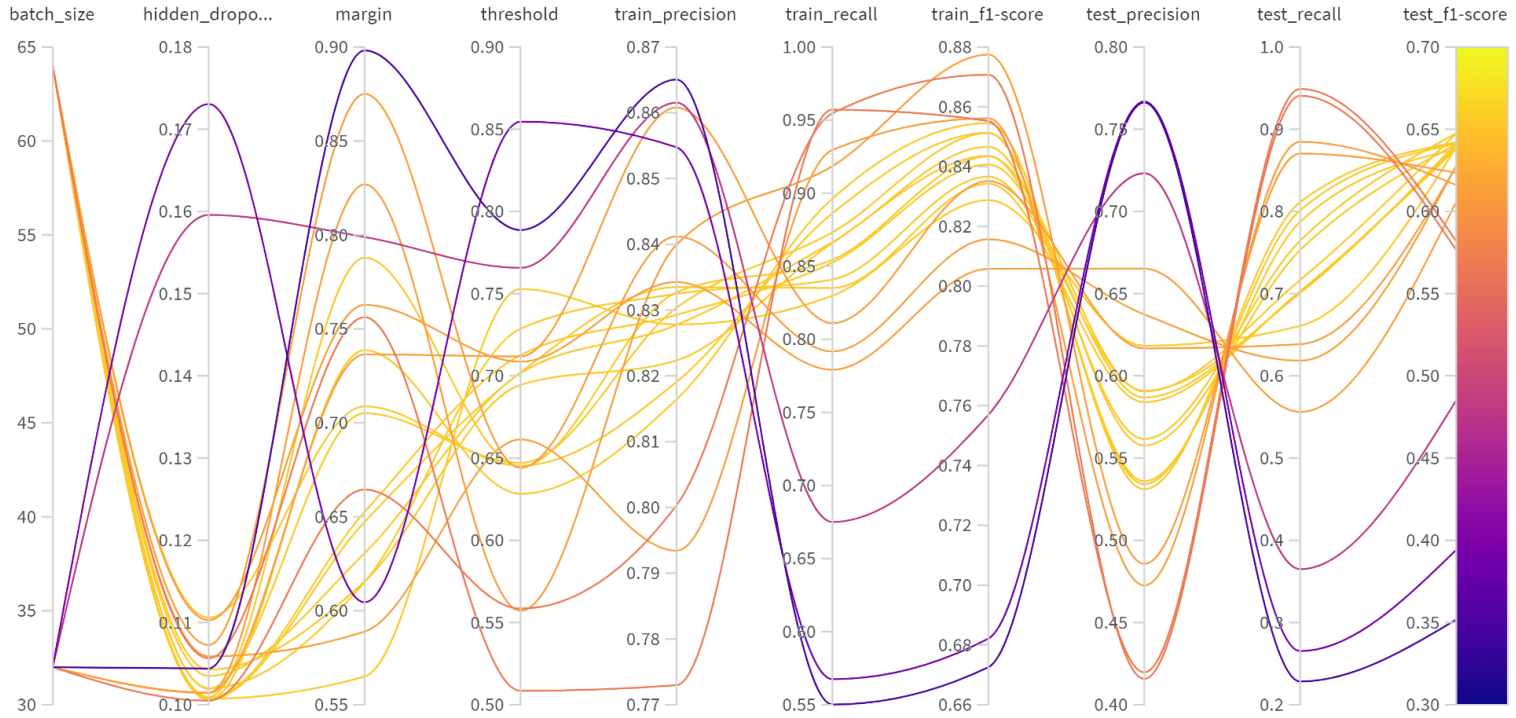
Pretrained Model	**Pretrained Model Architecure**	**Sentence Transformer Architecture**	**Data Augmentation Method And Hyperparamter Tuning**	**Loss Function**	**Precision**		**Recall**		**F1 Score**	
					train	**test**	**train**	**test**	**train**	**test**
all-MiniLM-L12-v2* (Trained on 1B+ sentence pairs)	BERT Based	Bi-Encoders	pn	. Constrastive Loss	0.70	0.33	0.99	1.0	0.82	0.49
			nb		0.68	0.39	0.99	0.98	0.81	0.50
			both pn + nb		0.52	0.34	0.99	0.99	0.68	0.50
all_datasets_v3_mpnet-base	MP-Net Based		pn		0.72	0.34	0.99	1.0	0.84	0.50
			nb		0.69	0.33	0.99	0.99	0.81	0.49
			Both pn + nb		0.58	0.34	0.99	0.99	0.74	0.51
paraphrase-mpnet-base-v2	MP-Net Based		Both pn + nb		0.54	0.33	0.99	0.99	0.70	0.55
all_datasets_v3_mpnet-base	MP-Net Based		None. Only Positive Examples	Multiple Negative Ranking Loss		0.61		0.72		0.66
all_datasets_v4_mpnet-base	MP-Net Based		None. Only Positive Examples			0.59		0.76		0.66
all_datasets_v3_MiniLM-L12	Bert Based		None. Only Positive Examples			0.60		0.74		0.66
all_datasets_v3_roberta-large	Bert Based		None. Only Positive Examples			0.61		0.77		0.68
paraphrase-distilroberta-base-v2	Bert Based		None. Only Positive Examples			0.59		0.63		0.61
stsb-roberta-large						0.53		0.73		0.61
distilroberta-base	Bert Based	Cross-Encoders	pn	Binary Cross Entropy Loss	0.94	0.44	0.97	0.89	0.96	0.58
			nb		0.90	0.33	0.83	0.77	0.87	0.46
			Both: pn +nb		0.74	0.37	0.76	0.46	0.75	0.41
stsb-roberta-large	Bert Based		pn		0.99	0.46	0.99	0.97	0.99	0.63
t5-large	Text to Text	Normal	pn	Cross Entropy Loss	0.99	0.48	0.99	0.97	0.99	0.65
flan-t5-base	Text to Text	Flan	pn		0.96	0.41	0.99	0.98	0.98	0.58
			pn dropout = 0.35		0.97	0.57	0.73	0.96	0.84	0.68
			pn+ nb dropout = 0.30		0.57	0.50	0.79	0.96	0.67	0.66
flan-t5-large	Text to Text	Flan	pn dropout = 0.35		0.75	0.36	0.95	0.99	0.85	0.53

Hyperparameter Tuning Bi-Encoders (Contrastive Loss) with Bayesian optimization in Wandb Sweep:

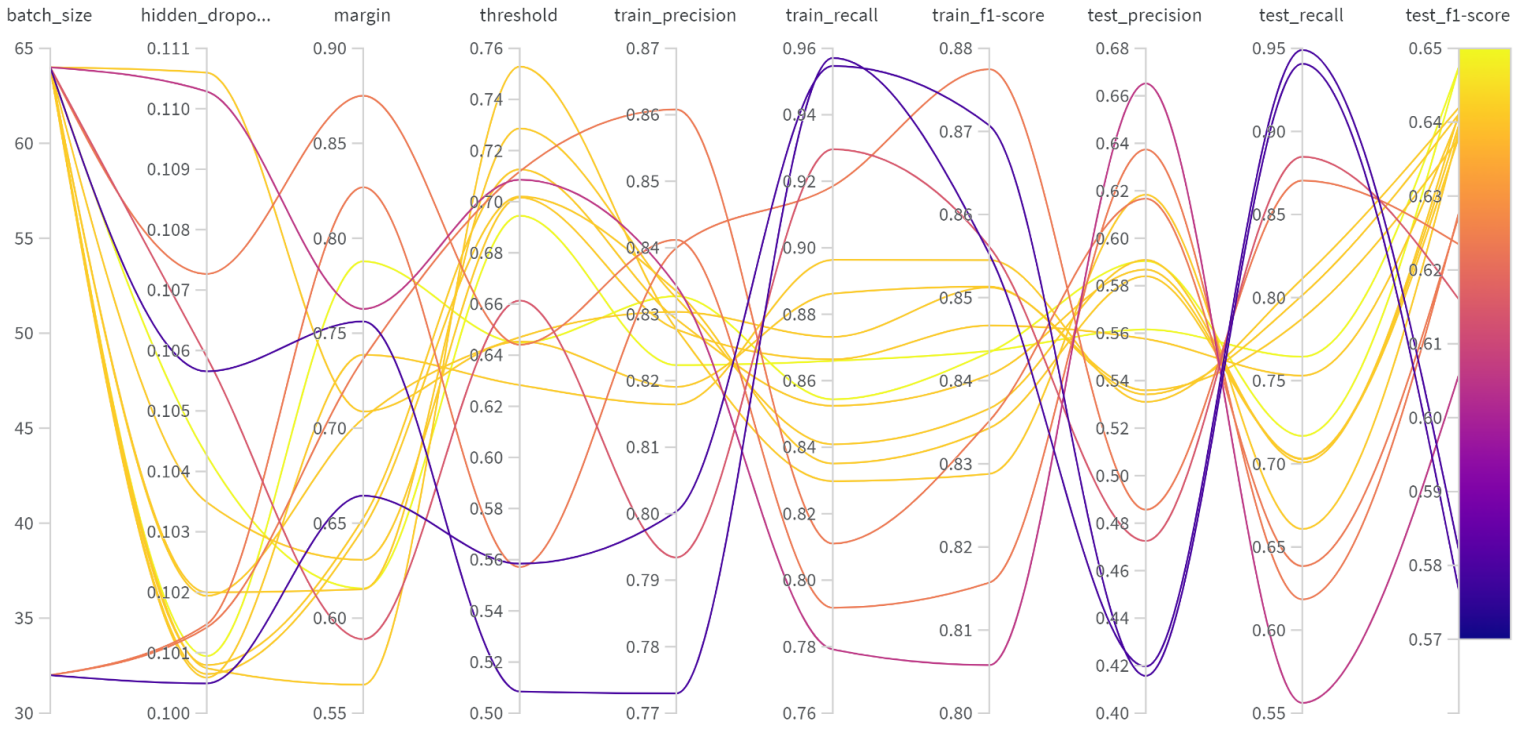
After conducting a thorough evaluation of various Contrastive Loss based Bi-Encoders, I have decided to use all_MiniLM-L12-v2 as our preferred model due to its good performance and faster training speed. When selecting the hyperparameters for this model, I carefully consider crucial parameters such as batch size, dropout rate, margin, and threshold. The margin parameter is especially important as it plays a significant role in the Contrastive Loss function. Furthermore, the threshold parameter, which determines the decision threshold between the cosine distance of two embeddings, must also be thoughtfully chosen to ensure optimal model performance. One method for optimizing hyperparameters is Bayesian optimization, which efficiently searches the hyperparameter space to find the optimal combination of parameters that yield the best results.
I opted to use Wandb for experiment tracking and Hyperparameter tuning, thanks to its remarkable Sweep feature. For training and tuning purposes, I utilized Kaggle GPU P100. Since we have a GPU limit on Kaggle, The Graphs and Plots images below were produced using Wandb.ai, and each title clearly explains the purpose of the corresponding feature. Please note that there are only 4 hyperparameters that are optimized.

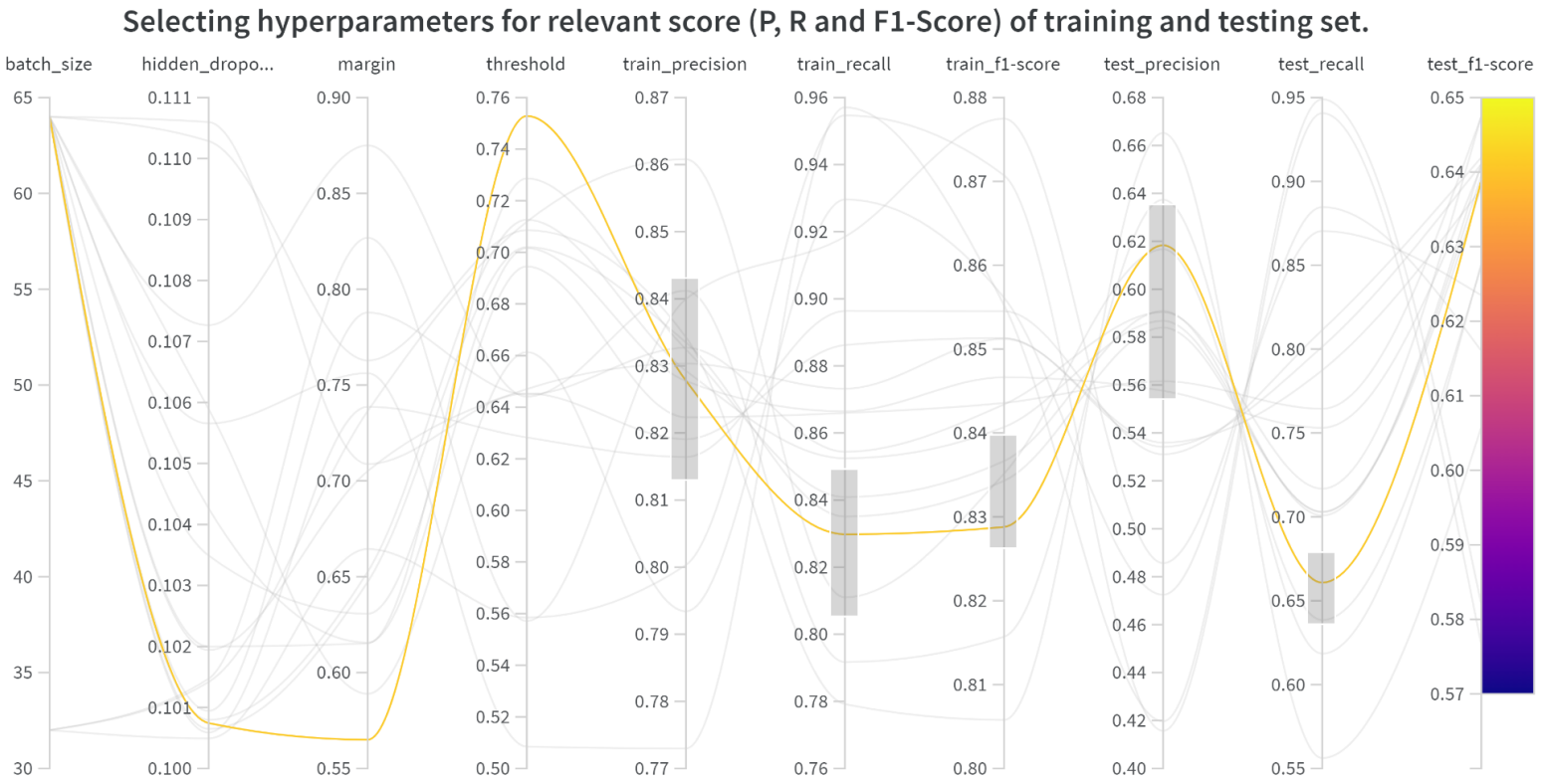


Hyperparameters vs Train and Test (Precision, Recall and F1-Score)



Zooming the section test_f1-score between [0.55 to 0.68]





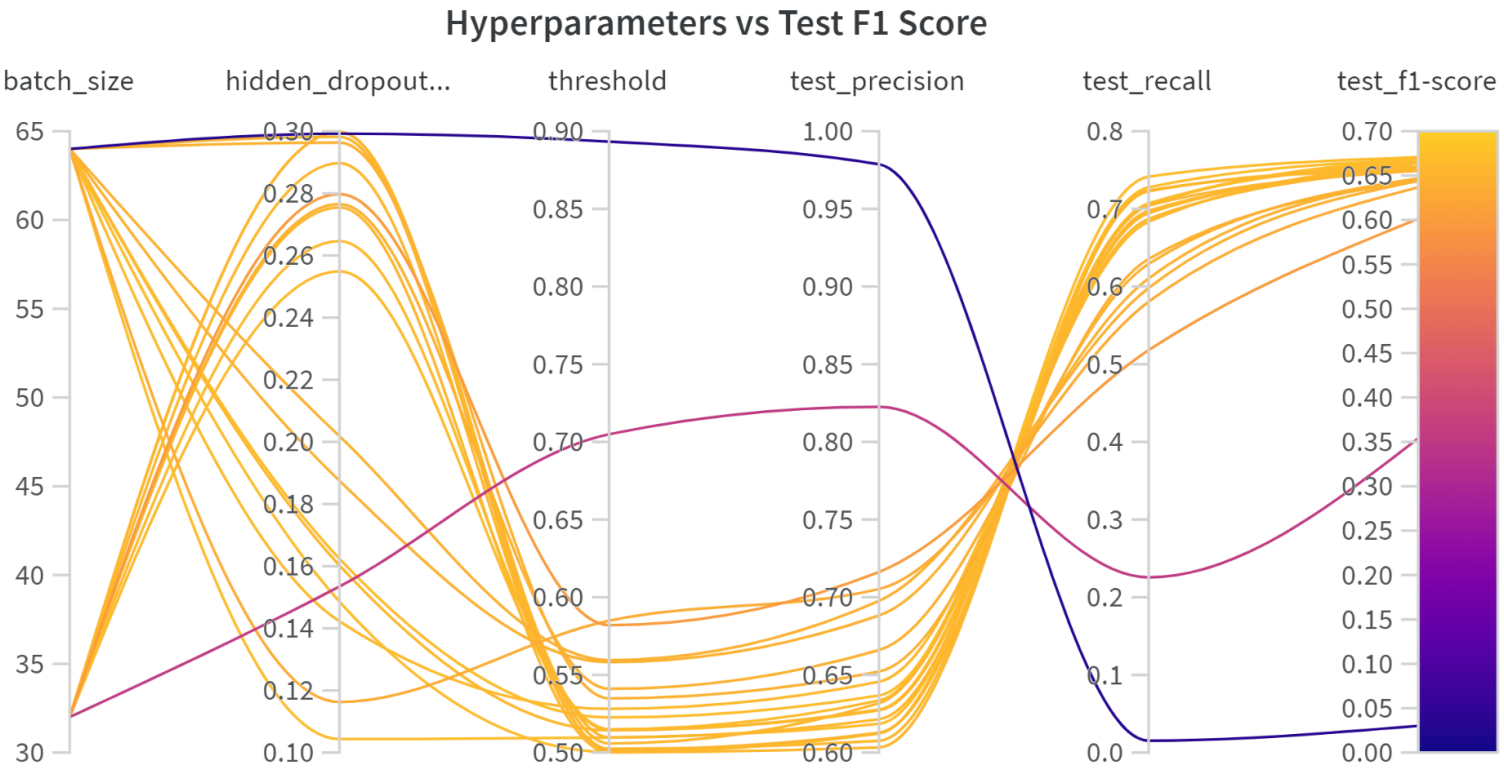
Selected Hyperparameters:-

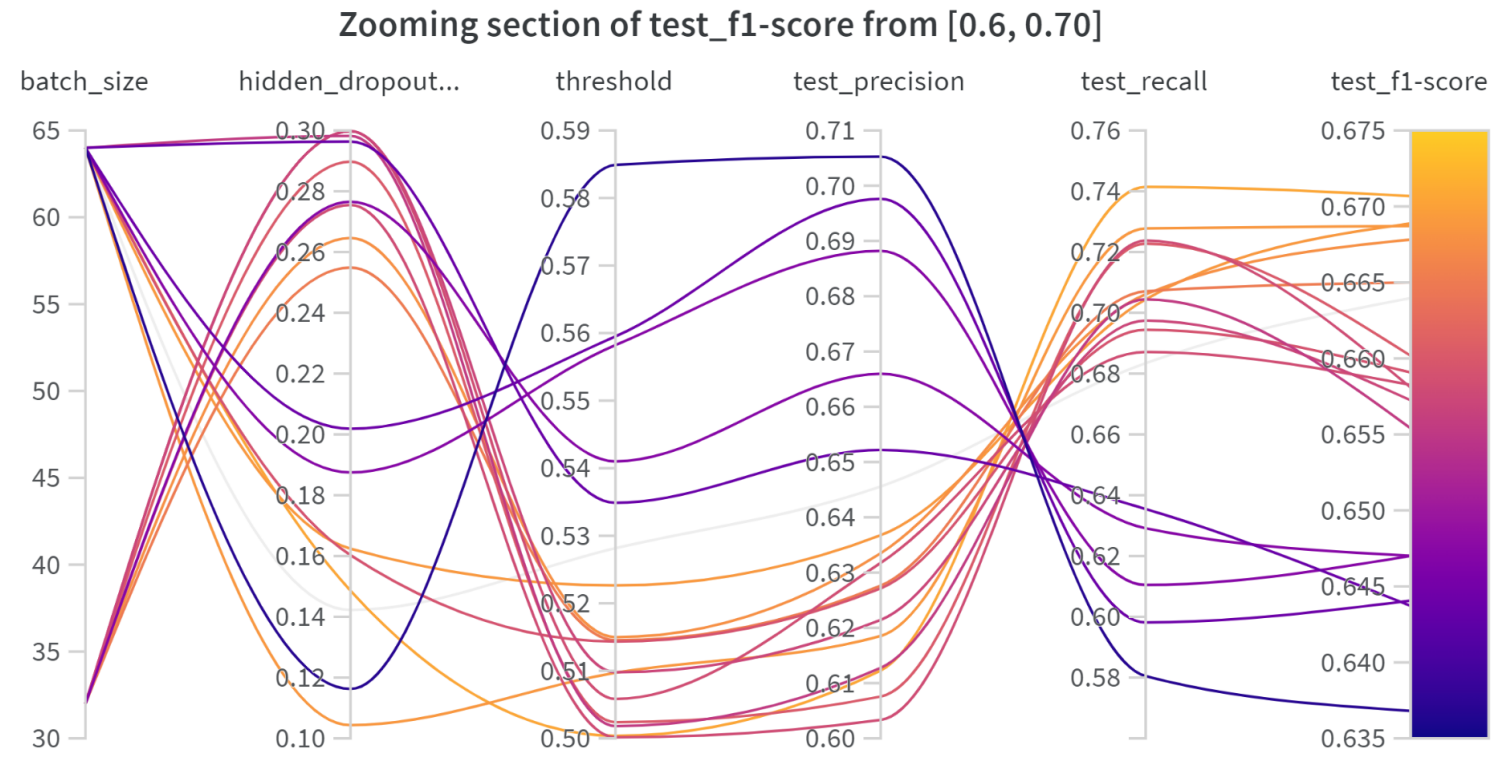
Hyperparameters	**Values of Selected Best Hyperparameters**	
hidden _dropout_prob	0.1007	
margin	0.565	
threshold	0.7528	
batch_size	64	
Scores	**Training Set**	**Testing Set**
Precision	0.8278	0.6183
Recall	0.8298	0.6608
F1-Score	0.8288	**0.6389**

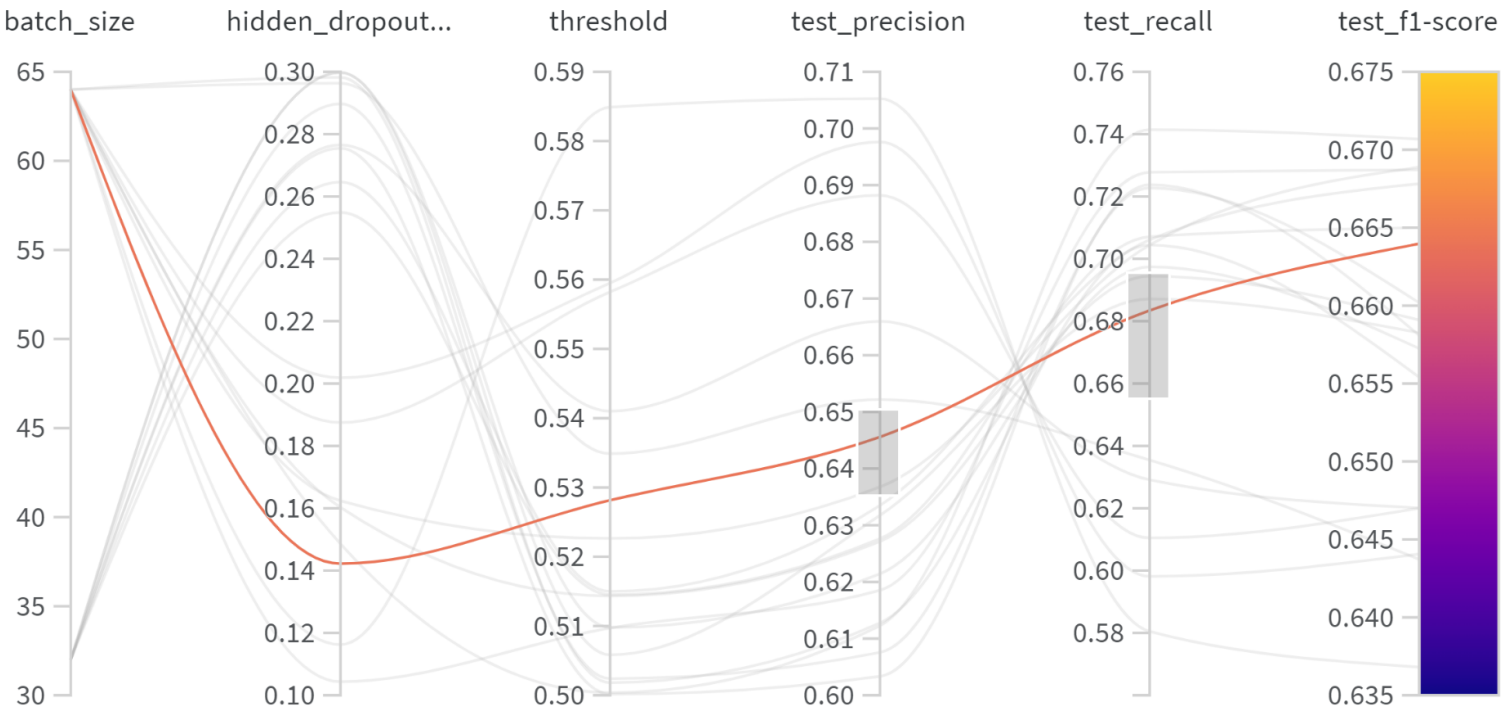
Hyperparameter Tuning Bi-Encoders (MNR Loss) with Bayesian optimization in Wandb Sweep:

In Bi-encoders with Multiple Negative Ranking (MNR) Loss, the margin hyperparameter is not included unlike Contrastive Loss, leaving only batch size, dropout, and threshold as the parameters that can be tuned. Despite this limitation, models using MNR loss have demonstrated exceptional performance on a single set of positive pairs, achieving the best scores so far. This approach has proved to be highly effective and has contributed significantly to

our overall success. The model all_datasets_v4_mpnet-base is trained on 1B+ on pretrained MP Net base (architecture `"MPNetForMaskedLM").`







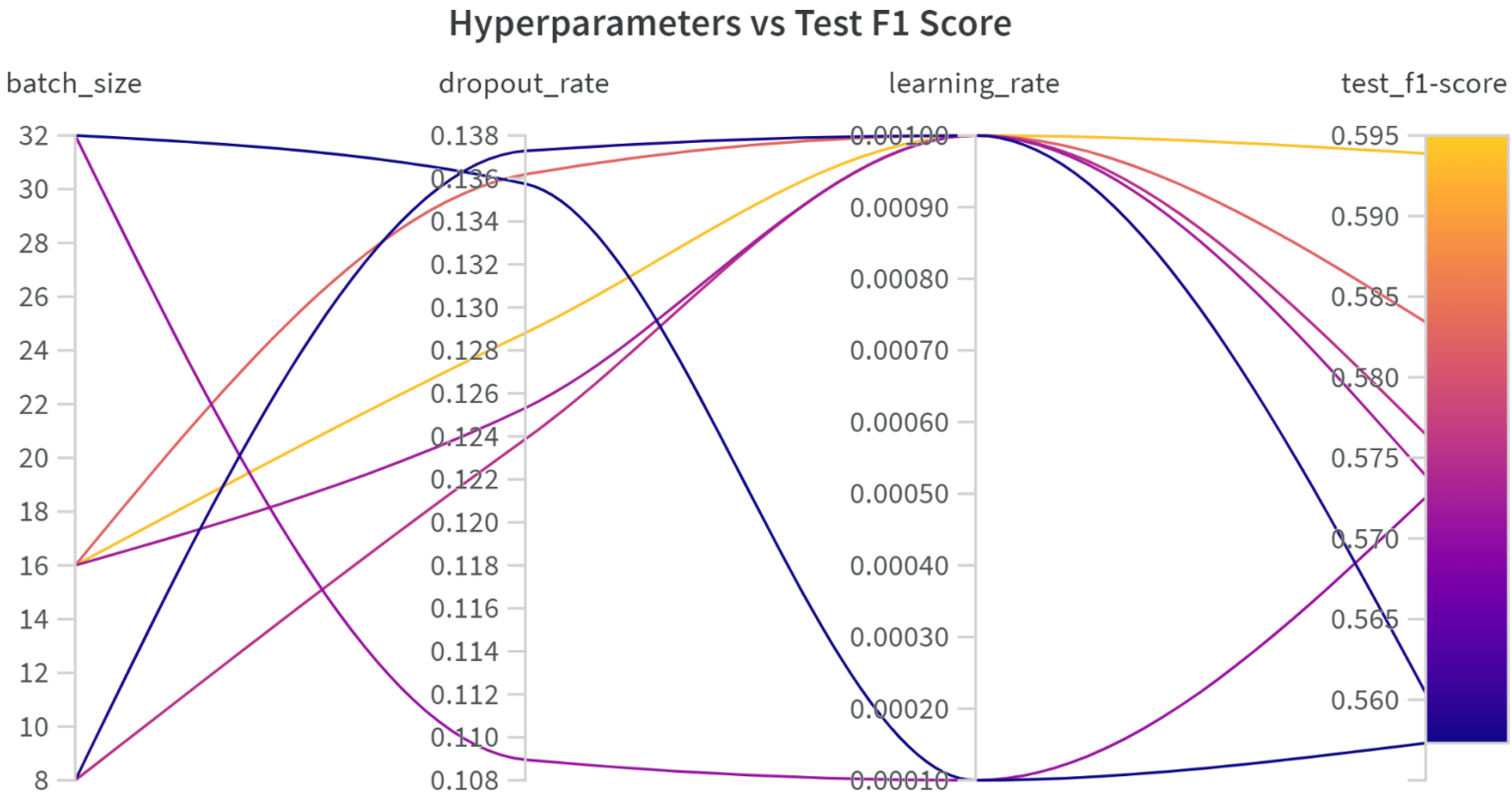
Hyperparameter Selected:-

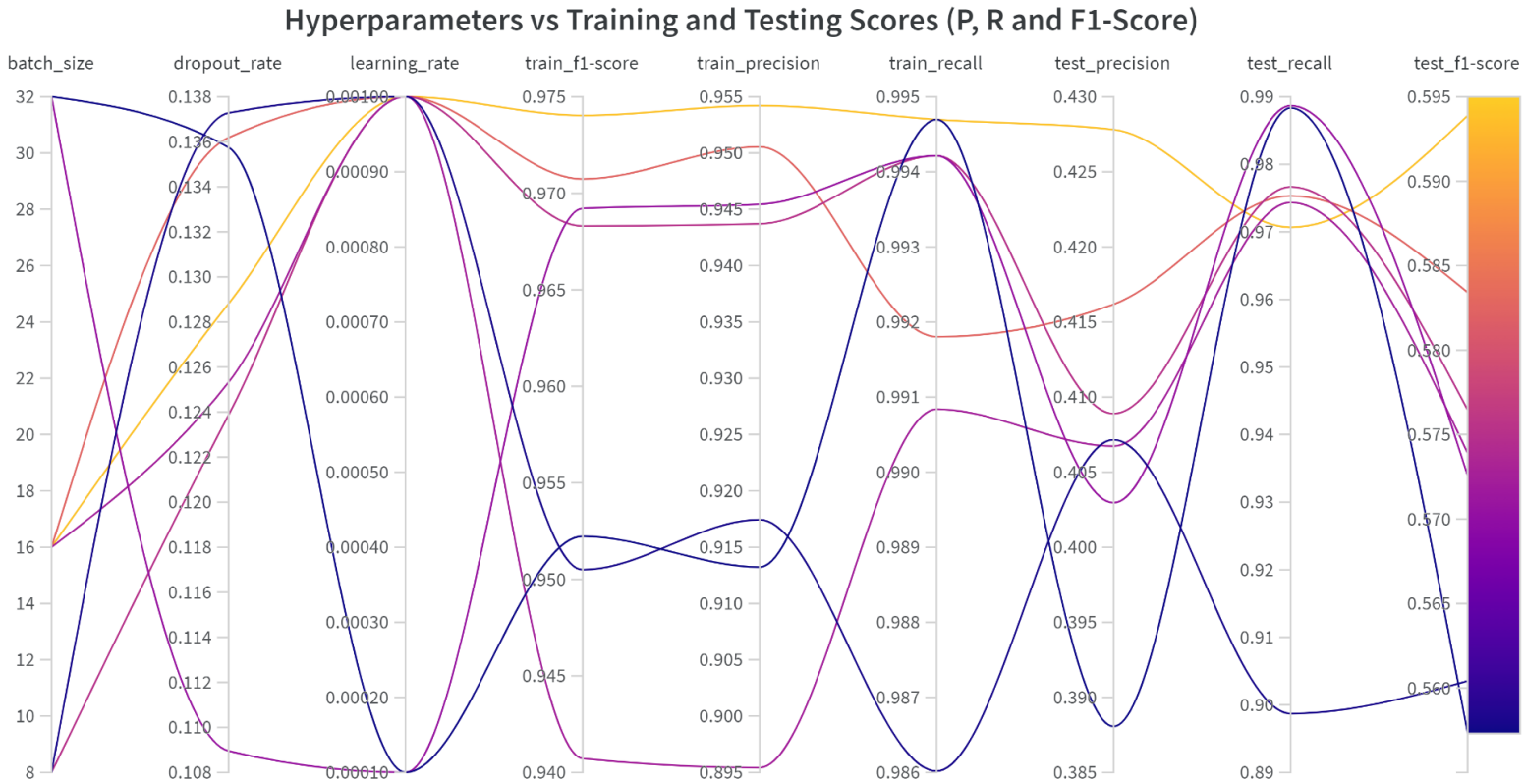
Hyperparameters Selected	Value for the Selected Model hyperparameter
Threshold	0.5282
Dropout prob	0.1422
Batch Size	64
Metrics	**Scores**
Test Precision Score	**0.6456**
Test Recall	**0.6834**
Test F1 Score	**0.664**

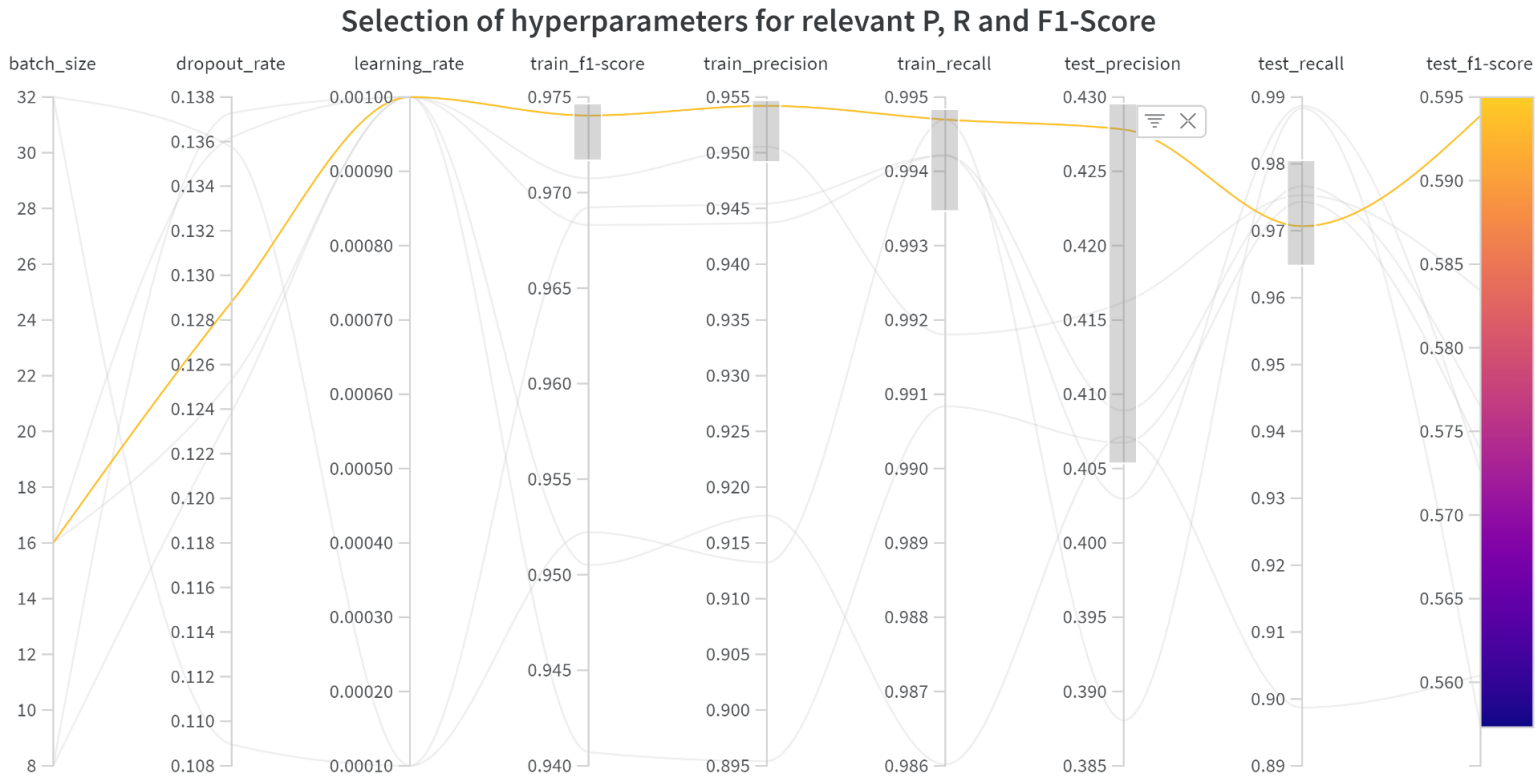
Hyperparameter Tuning Flan-t5-base with Bayesian optimization in Wandb Sweep:

I carefully considered several important hyperparameters, including batch size, learning rate, dropout rate, number of layers, and warmup-steps. After evaluating their impact on my model's performance, I decided to focus my efforts on tuning the learning rate, batch size, and dropout rate.I utilized the Adam optimizer, as it showed good results unless sgd based models generated wrong outputs other than “yes” or “no”. While other hyperparameters like number of layers and warmup-steps can also be important, I found that tuning these three specific hyperparameters had the greatest impact on my model's performance. The performance of the model lies below 0.6 for most of the

hyperparameters whereas bi-encoders has shown exceptional performance with less training time. So no need to select best hyperparameter for flan models.







Selected Hyperparameters:
The model is not good to compare to other models it shows very high recall scores compare to the precision score.

Hyperparameters Selected	Value for the Selected Model hyperparameter
Threshold	0.5282
Dropout prob	0.1288
Batch Size	16
Metrics	**Scores**
Test Precision Score	**0.42**
Test Recall	**0.97**
Test F1 Score	**0.5939**

Best Model

After fine-tuning bi-encoders with contrastive and Multiple Negative Ranking Loss and also leveraging FLAN based model on QA template for sentence pair classification. Bi -encoders with MNR Loss have shown exceptional performance on our downstream task with an f1-score of 0.664 on testing Set.

	all-MiniLM-L12-v2	all_datasets_v4_mpnet-base	flan-t5-base
Precision Score	0.6183	**0.6456**	0.42
Recall Score	0.6608	**0.6834**	0.97
F1 Score	0.6389	**0.664**	0.5939

Error Analysis

Confusion Matrix on Evaluation Set

		Actual	
		Positive	**Negative**
Prediction	**Positive**	**TP = 2099**	**FP = 1167**
	Negative	**FN = 902**	**TN = 4832**

Analysis of Number of words in Query and Reason

Type	**Query (“text”)**			**reason**			
No. of Words,	**min**	**max**	**mean**	**min**	**max**	**mean**	**No. of Unique Samples**
Evaluation	4	186	17.39	2	13	4.95	967 (100%)
True Positives	4	102	14.55	3	10	4.76	444 (45.9%)
False Positives	4	100	14.33	2	13	4.99	403 (41.6%)
True Negatives	4	186	18.25	2	13	5.06	723 (74.7%)
False Negatives	4	131	22.99	3	11	4.80	178 (18.4%)

Intersection between Unique No of Reasons between all metrics

Metrics	**tp**	**fp**	**tn**	**fn**
tp	444	208	297	138
fp	208	403	277	84
tn	297	277	723	148
fn	138	84	148	178

The model's behaviour does not exhibit any outliers or exceptions apart from the mean length of False Negatives and True Negatives, which are significantly higher compared to the others. This could potentially be one type of error in predicting negatives since both the True Negatives and False Negatives texts are relatively longer compared to the others.

1. False Positives:- False positives are incorrect results that indicate a positive outcome when in reality the actual result is negative. Randomly Selected 50 samples from 1167 samples of False Positives to analyze any errors.

---

a. Wrong Annotations (Labels that should be positive) Examples:-

---

Text:- not supportive of the arabic language
Reason:- want arabic language support
Text:- it has crashed at least a dozen times.
Reason:- app is crashing
Text:- this app is very useful for meetings
Reason:- good app for conducting online meeting
Text:- good but no new films how and when they will come
Reason:- need to update with new movies
Text:- i can not reinstall it i do not know why it was canceled
Reason:- want to reinstall the app
Text:- ik keep getting error 39.
Reason:- facing error 39 issue while using the app
Text:- and the fact that not every smart tv is also available as an app is also a huge shortcoming.
Reason:- want an app on smart tv
Text:- excellent app netflix 1 disney+ 2
Reason:- app service is better than netflix

---

b. Error where the “text” is too small to understand the reason

---

Text:- the shows are good but there are too many ads
Reason:- too many ads in app
Text:- lot of movies r not play
Reason:- missing some movies

---

c. Legit Errors are where the model doesn’t understand the meaning of query and reason most of the time this happens due to the length of the query.

Text:- zoom should have a 10 min option for meetings so that when you schedule one that you know a client isn't going to call in for you, you only waste 10 min instead of 15.
Reason:- want to increase meeting time limit
Text:- ever since disney bought espn, this app has steadily gone downhill. even with espn+, there are 15 or 30 second ads on every last video. and now a 33% price increase to espn+, this app should be 0 stars
Reason:- unable to watch espn+

---

2. **False Negatives:-** False Negative are incorrect results that indicate a negative outcome when in reality the actual result is Positive. Reason like “unable to use app” , “app is not good” have large occurrence in Negatively labelled data that’s why some time even if the reason satisfies the result is Negative.

Reason	No of time reason occurred in False Negative data.
unable to use app	115
app is not good	63

good for watching movies and serials	53
unable to login	38
unable to play offline videos	30

Conclusion:-

Based on the problem of Semantic Similarity in textual entailment, models such as sentence transformers have demonstrated outstanding performance. By utilizing Multiple Negative Ranking loss, it reduces the necessity of data augmentation for the Negative Label Generation task. After fine-tuning and hyperparameter tuning of the bi-encoder with MNR loss solely on Positive Examples, I accomplished an exceptional f1-score of 0.6634 for the sentence pair classification task.

References

1. Train and Fine-Tune Sentence Transformers Models | by Omar Espejel at HuggingFace
2. Constrastive Loss | Maksym Bekuzarov
3. Multiple Negative Ranking Loss | Nicholas Broad
4. Introducing FLAN: More generalizable Language Models with Instruction Fine-Tuning | by Maarten Bosma et al.
5. Finetune Language Models are zero shot learners Jason Wei et al.

Mistakes

1. Didn't used AUROC Metric. Perfect metrics for this problem.
2. Didn't used dev and test set because the data was less.