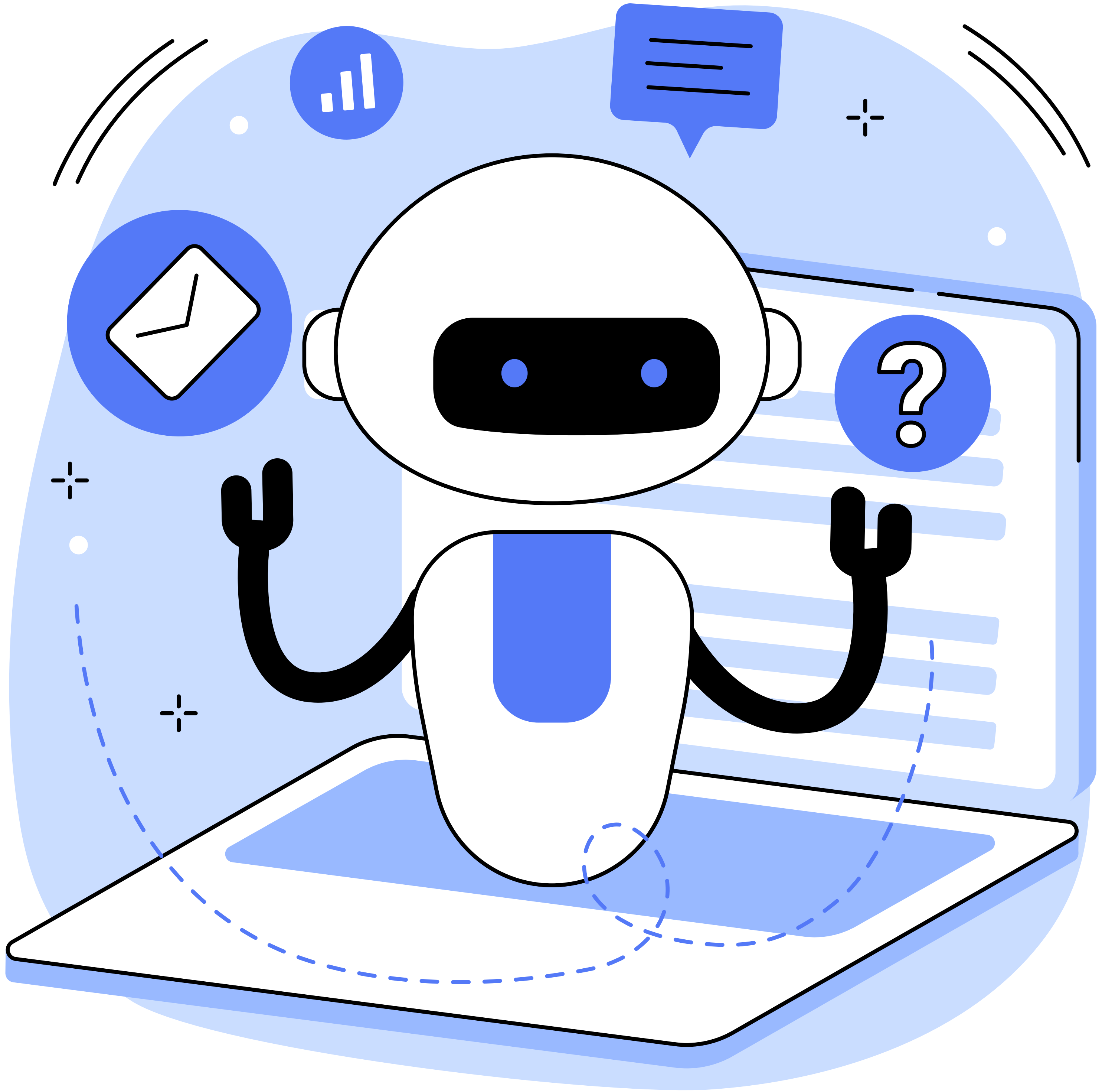
AUTOMATED RESEARCH PAPER CATEGORIZATION

KAMENG HOSTEL

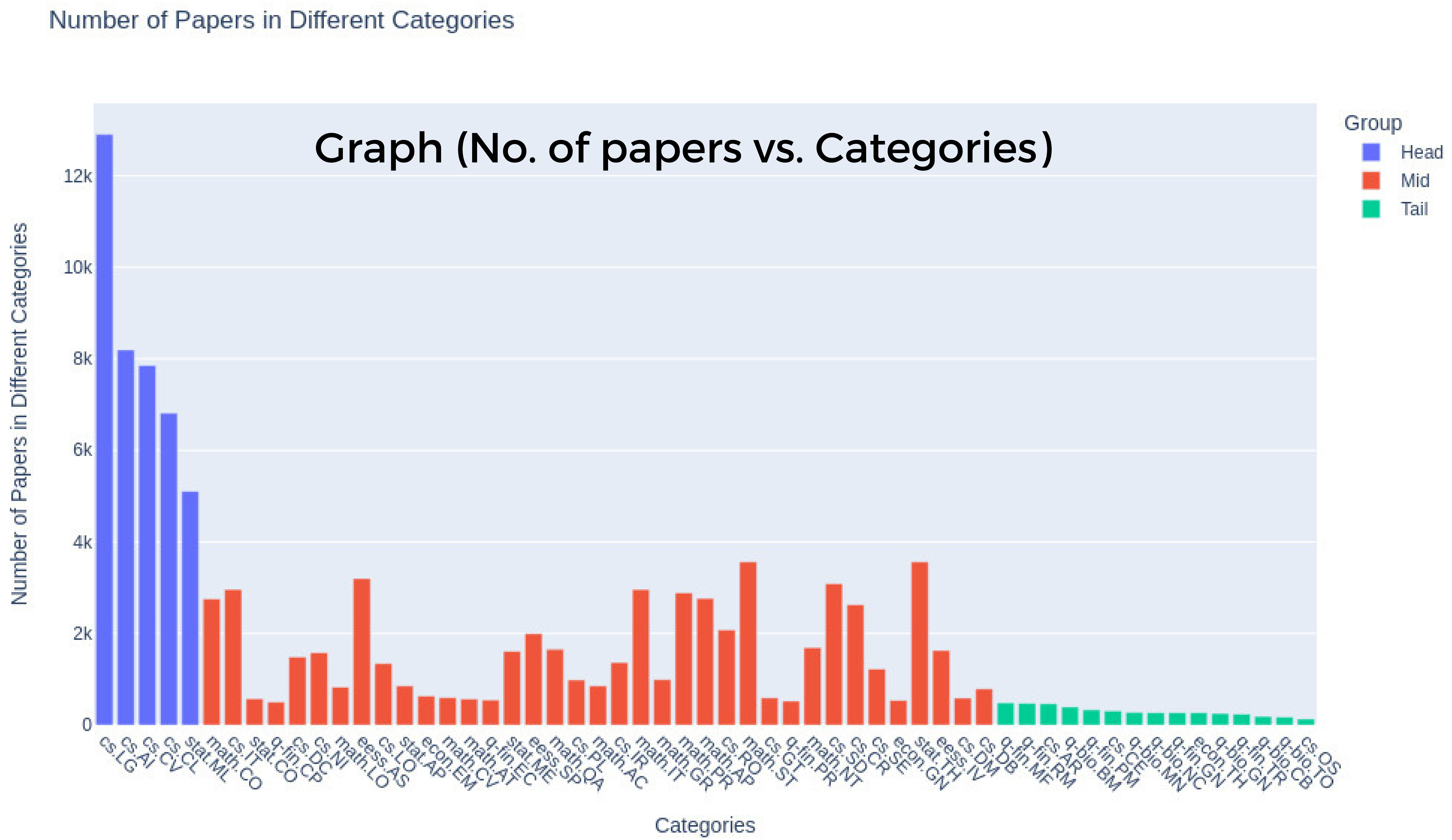
CONTENTS

OVERVIEW

Contemporary paper submission platforms necessitate users to upload paper titles and abstracts, followed by the selection of appropriate categories for their submissions. However, the multitude of available categories poses a challenge for authors seeking optimal classification. Imagine a submission system that not only streamlines this process but also enhances user experience by offering intelligent category suggestions based on the paper's content.



EXPLORATORY DATA ANALYSIS



EXPLORATORY DATA ANALYSIS

The length of Head is 5; The length of Mid is 37; The length of Tail is 15.

Through EDA we found that there are 23 rows with no information about the research paper. The following abstracts are:

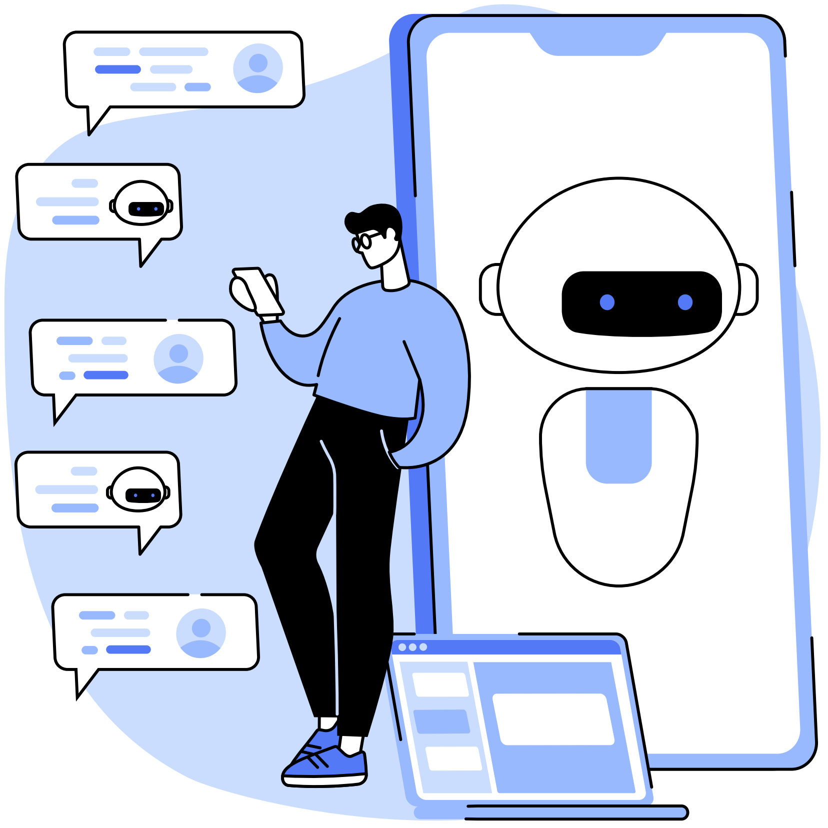
- 'No abstract available.': 6,
- 'This paper has been withdrawn': 5,
- 'This paper has been withdrawn by the author.': 4,
- 'This paper has been withdrawn by the author(s), due an error in the proof.': 3,
- 'This submission has been withdrawn at the request of the author.': 3,
- 'This paper has been withdrawn by the author; a revised version is part of the\nauthor\'s phd-thesis "Quasi-logarithmic structures" (Zurich, 2007).' : 2

STRATIFIED FOLD

```python
from skmultilearn.model_selection import iterative_train_test_split
import numpy as np
mlb = MultiLabelBinarizer()


def balanced_split(df, mlb, test_size=0.5):
    ind = np.expand_dims(np.arange(len(df)), axis=1)
    mlb.fit_transform(df['Categories'])
    labels = mlb.transform(df['Categories'])
    ind_train, _, ind_test, _ = iterative_train_test_split(
        ind, labels, test_size
    )
    return df.iloc[ind_train[:, 0]], df.iloc[ind_test[:, 0]]


df_train, df_tmp = balanced_split(df,mlb, test_size=0.2)
df_val, df_test = balanced_split(df_tmp,mlb, test_size=0.5)
```



MODEL JUSTIFICATION

Contextual Embeddings:

RoBERTa and DeBERTa use contextual embeddings, considering the entire context of a word in a sentence. This helps in capturing the relationships between words, which is crucial for understanding the context of each label in the context of the entire document. So, we did not text process the data.

Multi-Head Attention Mechanism:

The multi-head attention mechanism in transformer models, including RoBERTa and DeBERTa, allows the models to focus on different parts of the input sequence simultaneously. This is advantageous for capturing hierarchical relationships within the text, which is often present in multi-label text classification tasks.

Performance on Benchmarks:

RoBERTa and DeBERTa have demonstrated state-of-the-art performance on various NLP benchmarks, showcasing their effectiveness in capturing complex linguistic patterns and improving generalization across different tasks, including multi-label classification.

LOSS FUNCTION

Through EDA we can tell our training data is skewed. So, we used the Distributed Balanced loss function.

Model/ Loss Function	Reuters Total miF/maF	Reuters Head(≥35) miF/maF	Reuters Med(8-35) miF/maF	Reuters Tail(≤8) miF/maF	PubMed Total miF/maF	PubMed Head(≥50) miF/maF	PubMed Med(15-50) miF/maF	PubMed Tail(≤15) miF/maF
SVM	87.60/51.63	89.87/78.47	66.92/61.00	22.54/13.83	58.54/13.31	60.77/34.33	19.78/5.62	6.94/0.67
BCE	89.14/47.32	91.75/82.81	66.28/57.26	0.00/0.00	26.17/0.02	27.61/0.06	0.00/0.00	0.00/0.00
FL	89.97/56.83	91.83/82.64	76.16/70.63	27.40/15.37	58.30/13.94	60.43/33.69	26.39/8.15	8.58/0.86
CB	89.23/52.96	91.56/80.44	71.64/66.61	23.08/9.93	58.57/13.67	60.75/33.40	24.50/7.39	9.92/1.01
R-FL	89.47/54.35	91.59/80.39	72.86/66.69	25.00/14.22	57.90/14.66	59.85/34.09	30.32/9.70	11.45/1.15
NTR-FL	90.70/60.70	92.37/82.65	79.35/75.34	39.51/22.33	60.92/16.99	**63.15**/38.85	33.14/11.39	15.86/1.82
DB-0FL	89.45/57.98	91.21/82.05	77.33/71.11	31.17/19.05	58.95/15.15	60.99/34.92	31.06/10.02	14.23/1.49
CB-NTR	**90.74**/63.31	**92.46**/83.28	78.42/72.98	46.32/**32.31**	**61.07**/18.40	63.02/39.95	37.18/13.43	24.15/2.97
DB	90.62/**64.47**	92.14/**83.48**	**80.25/77.01**	**48.89**/31.39	60.63/**19.19**	62.39/**40.48**	**41.14/15.33**	**24.19/3.08**

We can observe that **DB** performed better than **BCE**

METRICS

	Precision	Recall	F1
Micro avg	0.76	0.75	0.75
Macro avg	0.72	0.72	0.71
Weighted avg	0.76	0.75	0.75
Samples avg	0.81	0.81	0.77

F1 Score

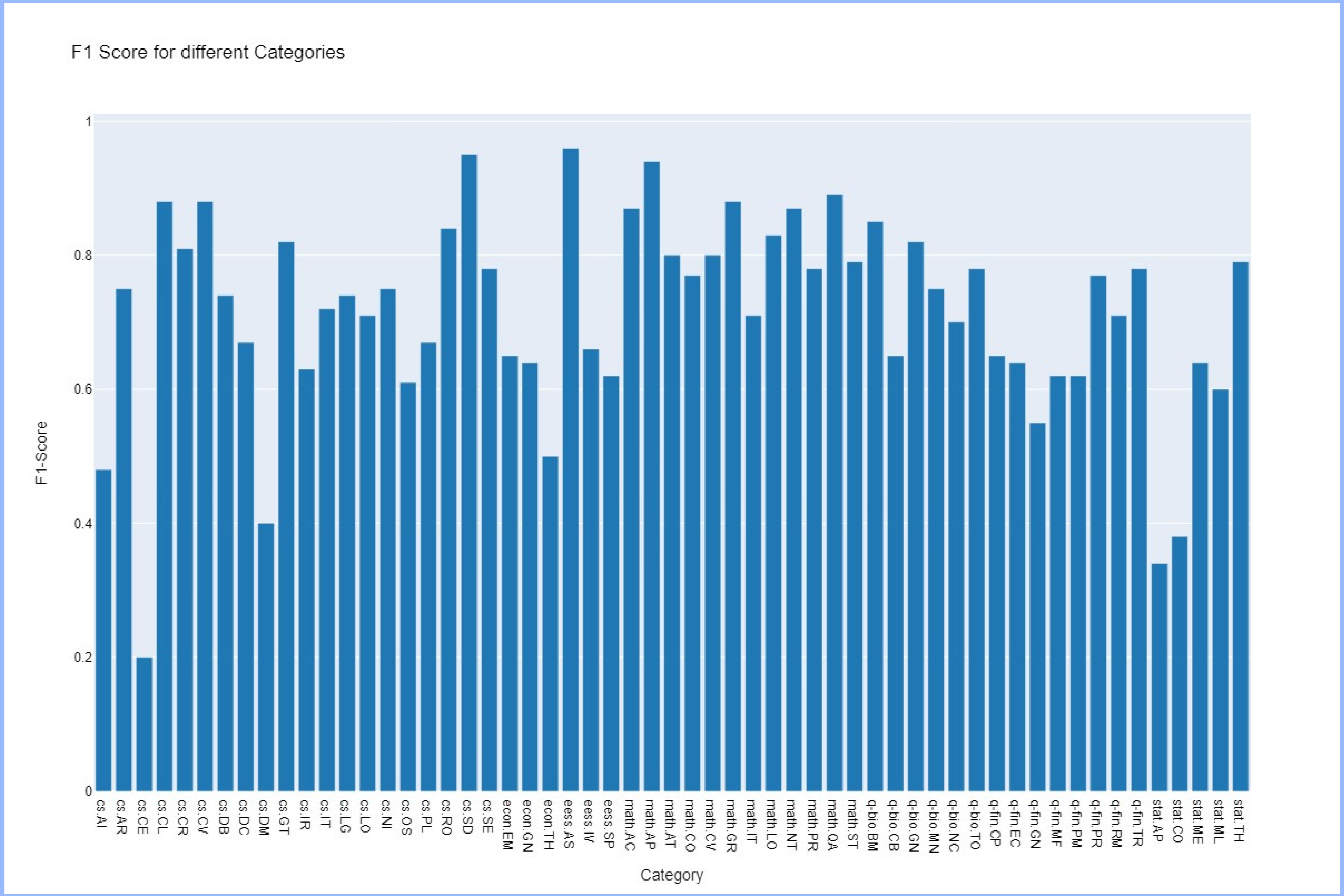
0.67

LB (Leader Board)

0.71

CV (Cross Validation)

METRICS


F1 Score for different Categories
F1-Score
0
0.2
0.4
0.6
0.8
1
Category
cs.AI
cs.AR
cs.CE
cs.CL
cs.CR
cs.CV
cs.DB
cs.DC
cs.DM
cs.GT
cs.IR
cs.IT
cs.LG
cs.LO
cs.NI
cs.OS
cs.PL
cs.RO
cs.SD
cs.SE
econ.EM
econ.GN
econ.TH
eess.AS
eess.IV
eess.SP
math.AC
math.AP
math.AT
math.CO
math.CV
math.GR
math.IT
math.LO
math.NT
math.PR
math.QA
math.ST
q-bio.BM
q-bio.CB
q-bio.GN
q-bio.MN
q-bio.NC
q-bio.TO
q-fin.CP
q-fin.EC
q-fin.GN
q-fin.MF
q-fin.PM
q-fin.PR
q-fin.RM
q-fin.TR
stat.AP
stat.CO
stat.ME
stat.ML
stat.TH

HOW CAN WE HELP STUDENTS USE AI?

As **Language Models** improve, it can be helpful for EFL teachers to adopt these tools and help their students learn how to use them effectively. Here are some practical ideas.

FUNCTIONS

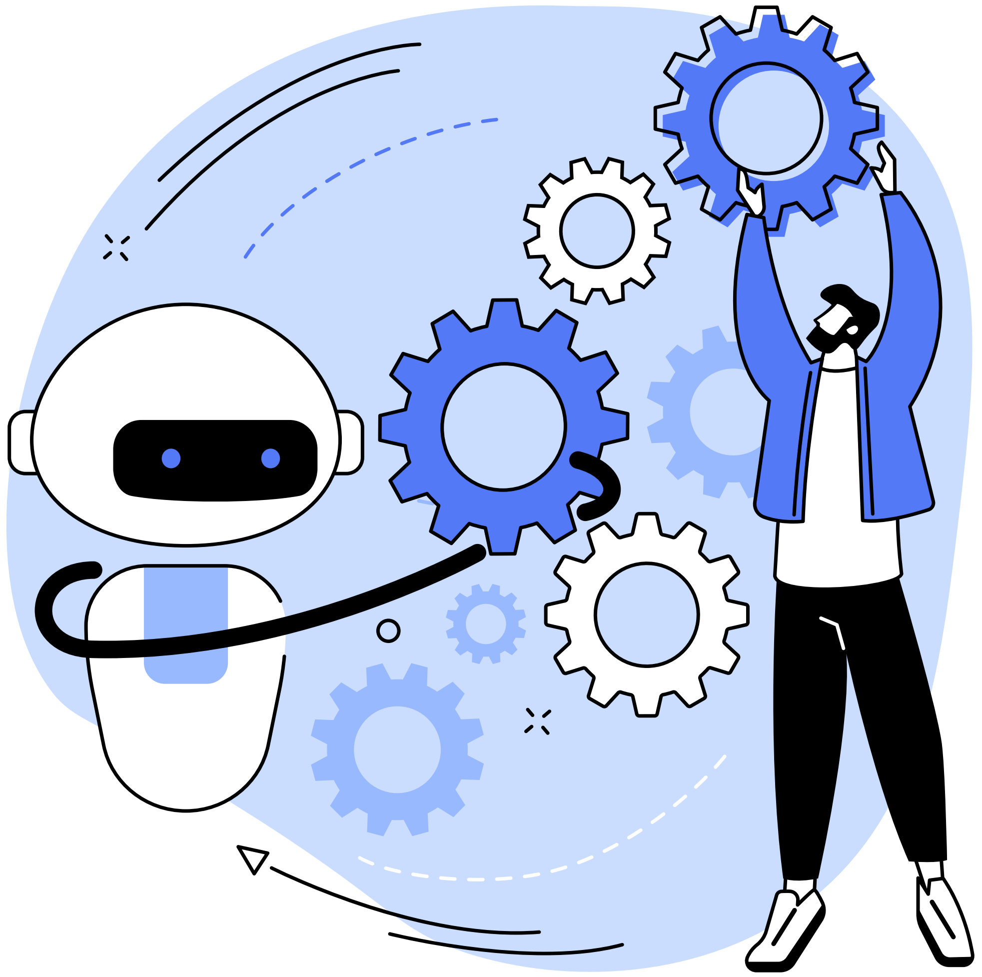
```python
def treatingzero_cases(predictions,prediction_probs):
    all_zeros_mask = torch.all(predictions == 0, dim=1)
    rows_with_all_zeros = torch.nonzero(all_zeros_mask).squeeze()
    for idx in rows_with_all_zeros:
        idx_value=torch.argmax(prediction_probs[idx,:])
        predictions[idx,idx_value]=1
    return predictions
```

Function: treatingzero_cases

```python
def optimise_f1_score(true_labels: np.ndarray, pred_labels: np.ndarray):
    best_med_th = 0.5
    true_bools = [tl == 1 for tl in true_labels]
    micro_thresholds = (np.array(range(-45, 15)) / 100) + best_med_th
    f1_results, prec_results, recall_results = [], [], []
    for th in micro_thresholds:
        pred_bools = [pl > th for pl in pred_labels]
        test_f1 = f1_score(true_bools, pred_bools, average="macro", zero_division=0)
        test_precision = precision_score(
            true_bools, pred_bools, average="macro", zero_division=0
        )
        test_recall = recall_score(
            true_bools, pred_bools, average="macro", zero_division=0
        )
        f1_results.append(test_f1)
        prec_results.append(test_precision)
        recall_results.append(test_recall)
        best_f1_idx = np.argmax(f1_results)
    return micro_thresholds[best_f1_idx]
```

Function: optimise_f1_score

PIPELINE

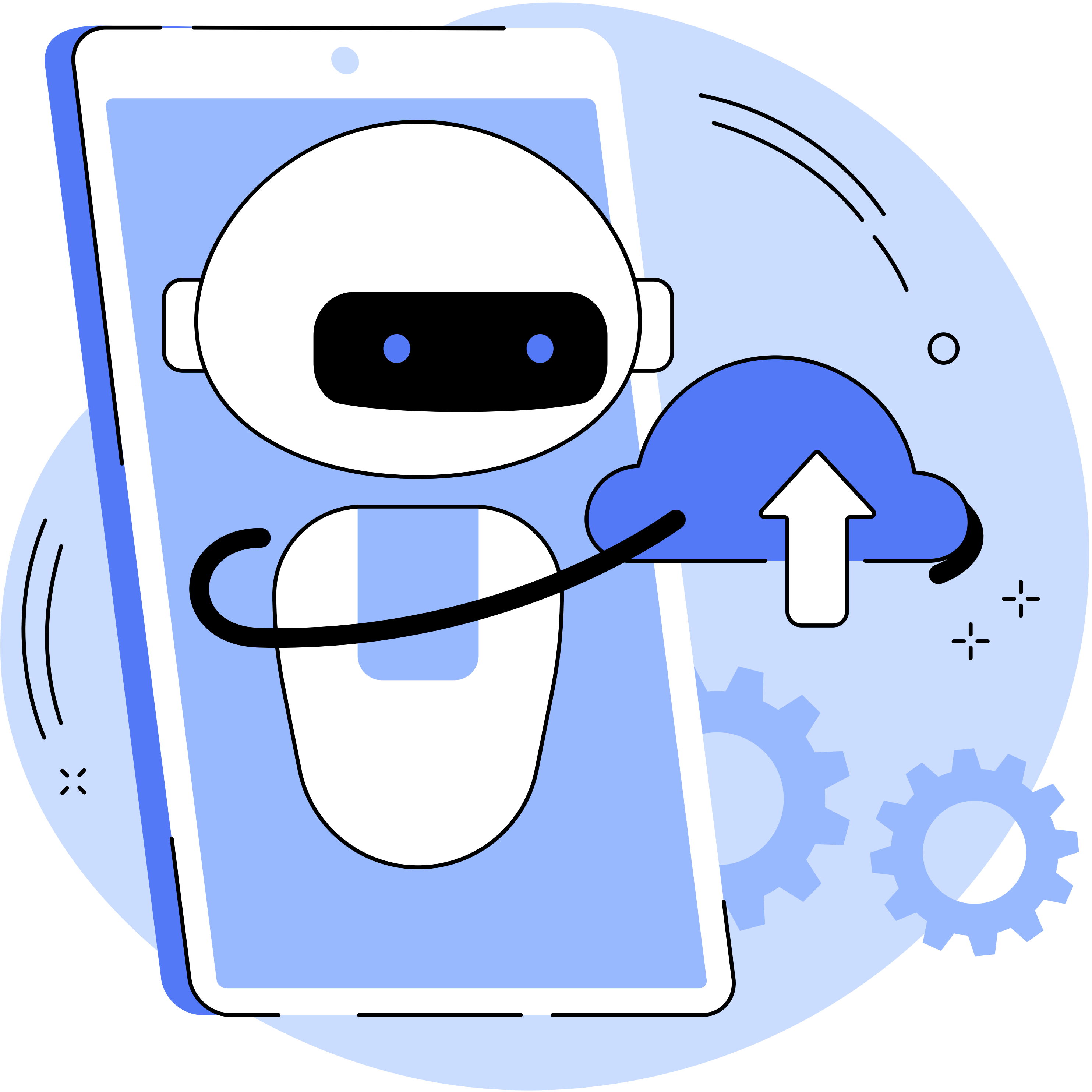


Models

- **Deberta-base**
- **Deberta-large**
- **Deberta-base**
- **Roberta-base**

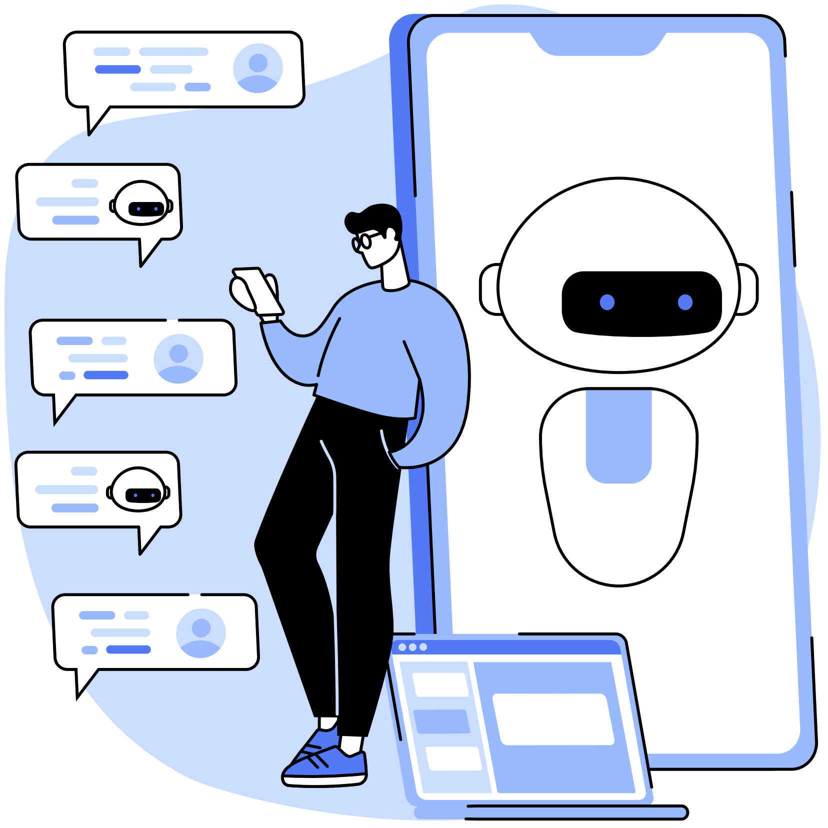
FUTURE ASPECTS

1. Cluster-based learning
2. Student-teacher model
3. Quantization
4. Pruning of model



REFERENCES

1. Roberta Ref.- https://arxiv.org/abs/1907.11692
2. Deberta Ref.- https://arxiv.org/abs/2006.03654
3. DB Loss Ref.- https://arxiv.org/abs/2109.04712



THANK YOU FOR LISTENING!