Report

Project Title: Rating Quality of Discourse Elements in Arguments

Course Name: Natural Language Processing (CS458)

Team Members:

Anant Terkar (20BCS014)

Yash Nikam (20BCS093)

Om Morendha (20BCS095)

Rahul Pentamsetty (20BCS106)

Under the guidance of:

Dr. Chinmayananda A

And

Dr. Nataraj K S

Aim:

This project aims at developing an automated feedback system for school students to help them shape their argumentative essay writing skills. An automated feedback tool is one way to make it easier for teachers to grade writing tasks assigned to their students which will also improve their writing skills.

Introduction:

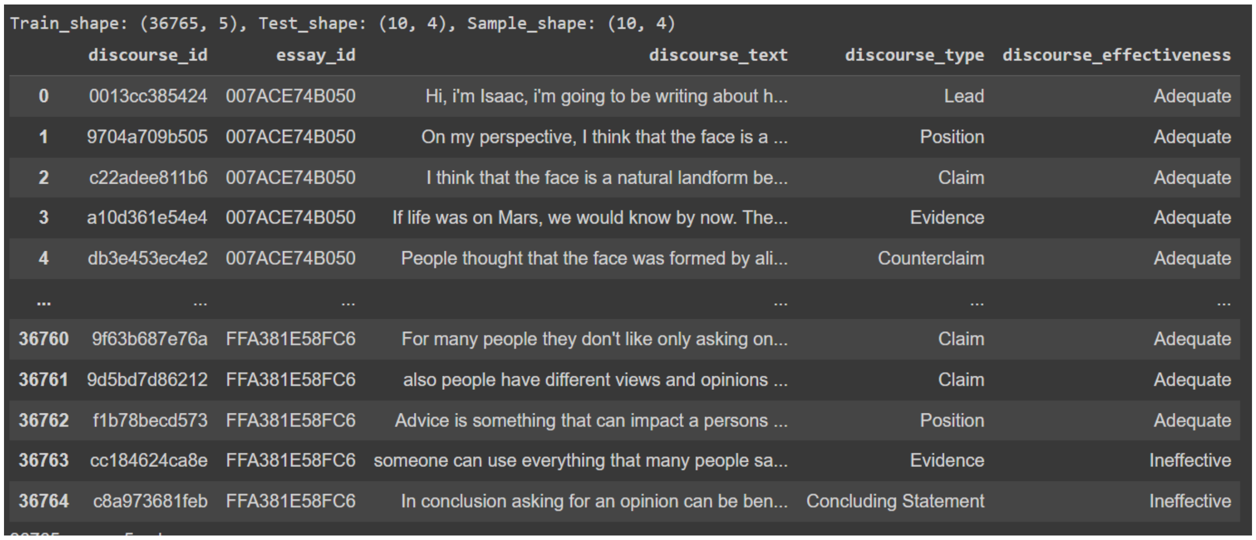
Every argumentative essay is divided into seven discourse elements, which are

- Lead: an opening that uses a fact, a quote, a description, or any other tactic to draw the reader in and highlight the thesis.
- Position: an opinion or conclusion on the main question
- Claim: a claim that backs the position.
- Counterclaim: a claim that disputes another statement or offers an argument against the stance.
- Rebuttal: a claim that challenges the counterclaim.
- Evidence: ideas or examples that support claims, counterclaims, or rebuttals.
- Concluding Statement: a concluding statement that restates the claims.

Each of these discourse elements is quality rated into three categories, ‘Ineffective’, ‘Adequate’, and ‘Effective’.

Dataset:

The dataset is taken from Kaggle[1]. The dataset consists of a .csv file containing the annotated discourse elements of each essay, including the quality ratings. Certain parts of the essay do not come under any of the above-annotated discourse elements. Such parts of the essay are removed.



```
Train_shape: (36765, 5), Test_shape: (10, 4), Sample_shape: (10, 4)
```

	discourse_id	essay_id	discourse_text	discourse_type	discourse_effectiveness
0	0013cc385424	007ACE74B050	Hi, i'm Isaac, i'm going to be writing about h...	Lead	Adequate
1	9704a709b505	007ACE74B050	On my perspective, I think that the face is a ...	Position	Adequate
2	c22adee811b6	007ACE74B050	I think that the face is a natural landform be...	Claim	Adequate
3	a10d361e54e4	007ACE74B050	If life was on Mars, we would know by now. The...	Evidence	Adequate
4	db3e453ec4e2	007ACE74B050	People thought that the face was formed by ali...	Counterclaim	Adequate
...	...	...	...	...	...
36760	9f63b687e76a	FFA381E58FC6	For many people they don't like only asking on...	Claim	Adequate
36761	9d5bd7d86212	FFA381E58FC6	also people have different views and opinions ...	Claim	Adequate
36762	f1b78becd573	FFA381E58FC6	Advice is something that can impact a persons ...	Position	Adequate
36763	cc184624ca8e	FFA381E58FC6	someone can use everything that many people sa...	Evidence	Ineffective
36764	c8a973681feb	FFA381E58FC6	In conclusion asking for an opinion can be ben...	Concluding Statement	Ineffective

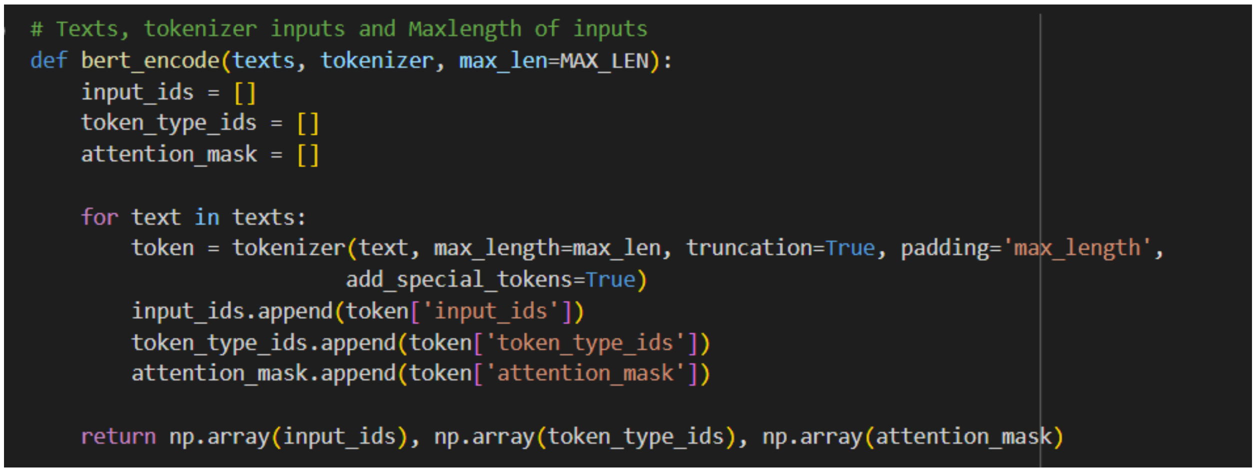
Overview of Dataset

We performed essential data cleaning such as removing stop words and apostrophes.

Models and Algorithms:

We have used the Bert encoder for encoding the discourse text. There were multiple approaches used to solve this problem. The first approach was to build a simple neural network on top of the transformer.

The following is the encoding function:

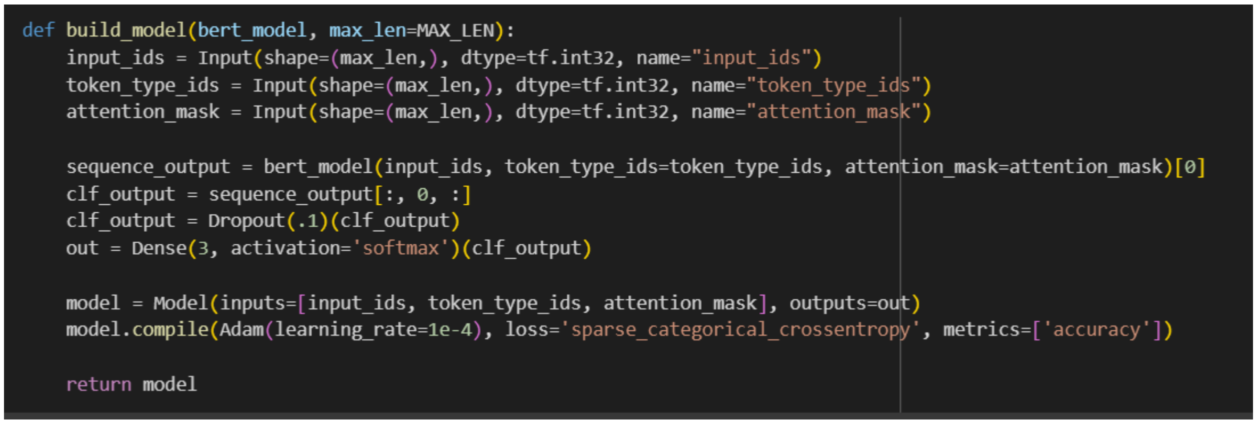


```
# Texts, tokenizer inputs and Maxlength of inputs
def bert_encode(texts, tokenizer, max_len=MAX_LEN):
    input_ids = []
    token_type_ids = []
    attention_mask = []

    for text in texts:
        token = tokenizer(text, max_length=max_len, truncation=True, padding='max_length',
                          add_special_tokens=True)
        input_ids.append(token['input_ids'])
        token_type_ids.append(token['token_type_ids'])
        attention_mask.append(token['attention_mask'])

    return np.array(input_ids), np.array(token_type_ids), np.array(attention_mask)
```

This code is for building the model:



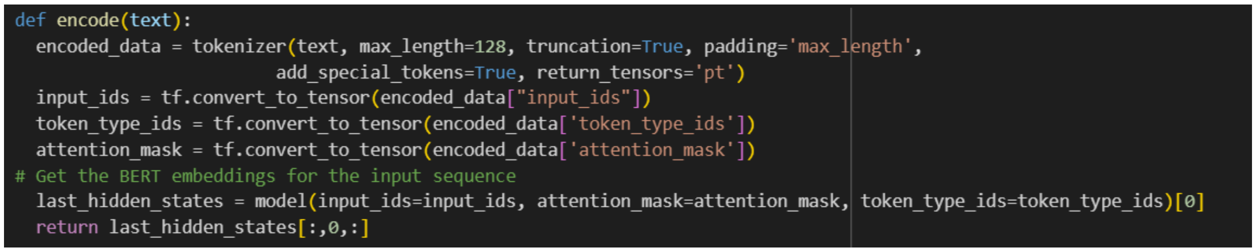
```
def build_model(bert_model, max_len=MAX_LEN):
    input_ids = Input(shape=(max_len,), dtype=tf.int32, name="input_ids")
    token_type_ids = Input(shape=(max_len,), dtype=tf.int32, name="token_type_ids")
    attention_mask = Input(shape=(max_len,), dtype=tf.int32, name="attention_mask")

    sequence_output = bert_model(input_ids, token_type_ids=token_type_ids, attention_mask=attention_mask)[0]
    clf_output = sequence_output[:, 0, :]
    clf_output = Dropout(.1)(clf_output)
    out = Dense(3, activation='softmax')(clf_output)

    model = Model(inputs=[input_ids, token_type_ids, attention_mask], outputs=out)
    model.compile(Adam(learning_rate=1e-4), loss='sparse_categorical_crossentropy', metrics=['accuracy'])

    return model
```

The next step was to extract embeddings from the BERT model so that we could apply custom classifiers to the embeddings:

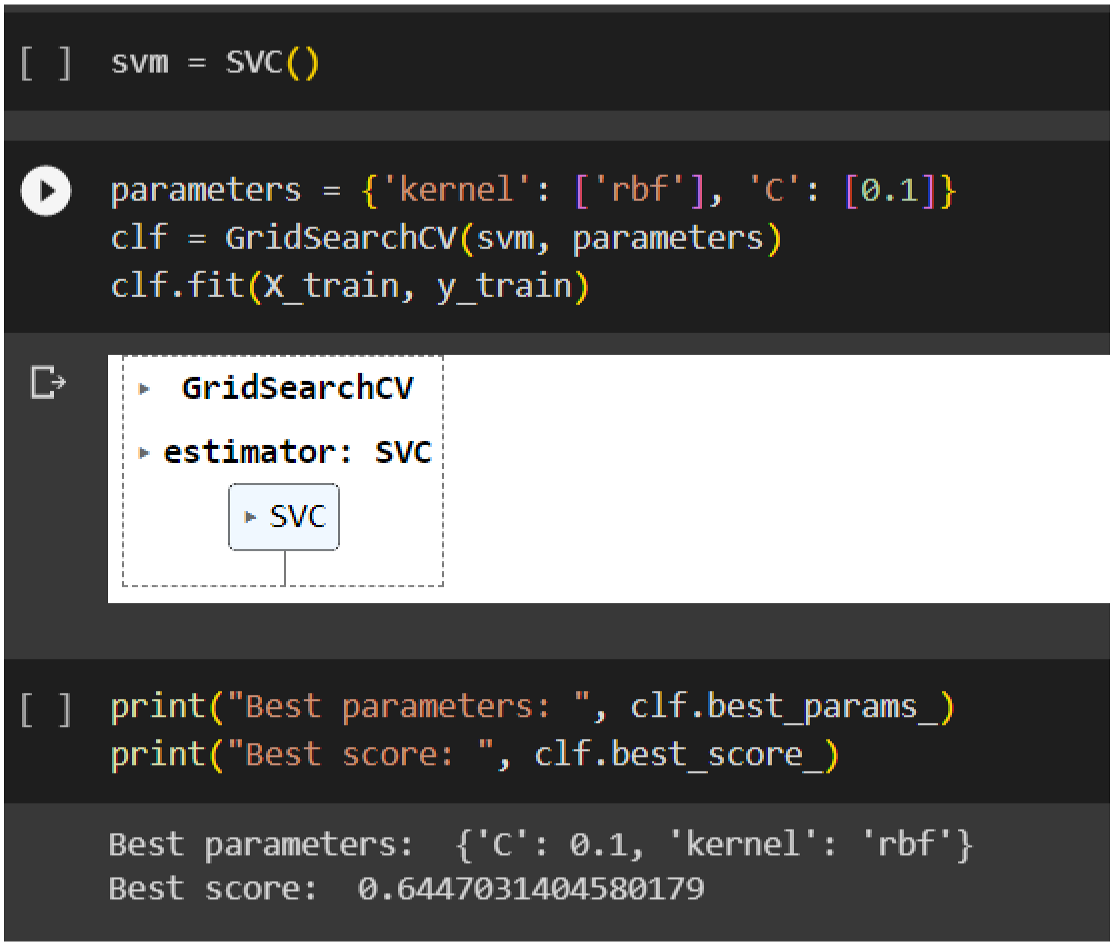


```
def encode(text):
  encoded_data = tokenizer(text, max_length=128, truncation=True, padding='max_length',
                           add_special_tokens=True, return_tensors='pt')
  input_ids = tf.convert_to_tensor(encoded_data["input_ids"])
  token_type_ids = tf.convert_to_tensor(encoded_data['token_type_ids'])
  attention_mask = tf.convert_to_tensor(encoded_data['attention_mask'])
# Get the BERT embeddings for the input sequence
  last_hidden_states = model(input_ids=input_ids, attention_mask=attention_mask, token_type_ids=token_type_ids)[0]
  return last_hidden_states[:,0,:]
```

Once the embeddings were stored in a CSV file. The following classifiers were used:

1. **Support Vector Classifier:**

   Initially, multiple parameters were passed in the GridSearch and then the best parameters were taken.



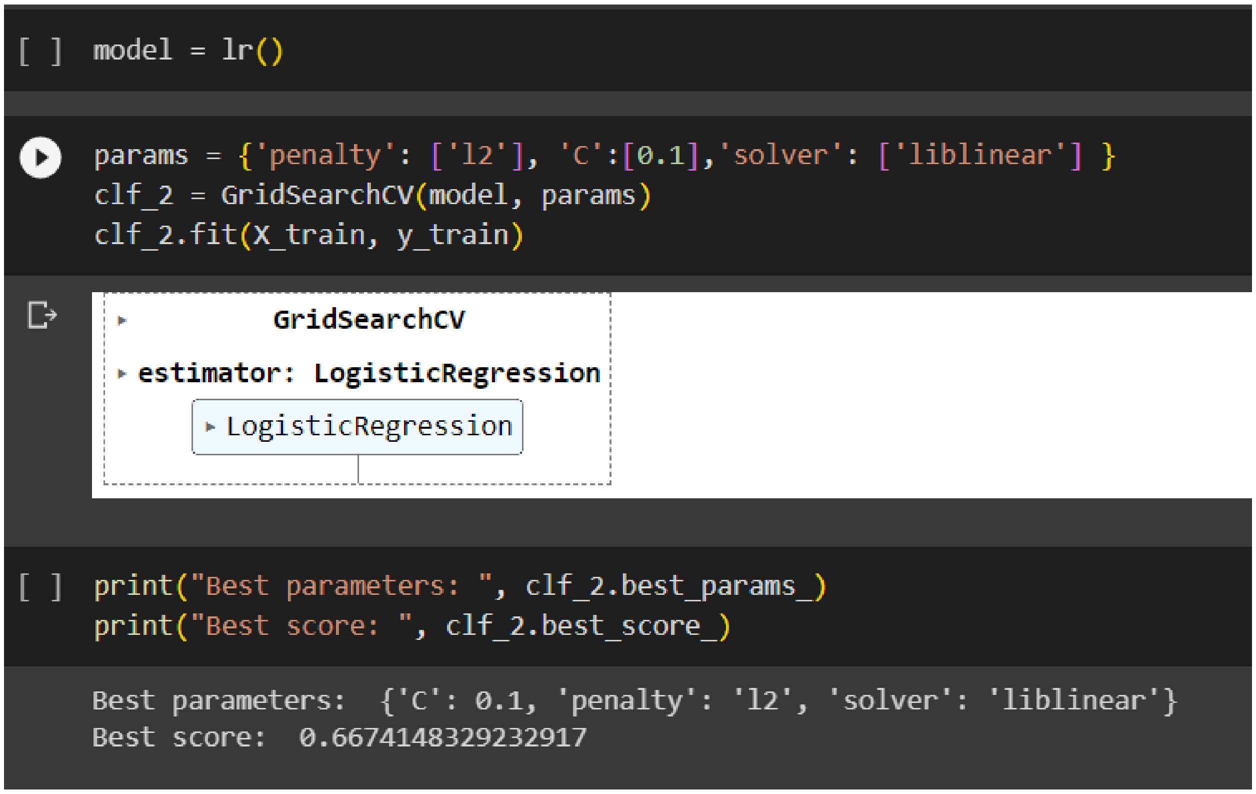
```
[ ] svm = SVC()
```

```
parameters = {'kernel': ['rbf'], 'C': [0.1]}
clf = GridSearchCV(svm, parameters)
clf.fit(X_train, y_train)
```

```
[ ] print("Best parameters: ", clf.best_params_)
    print("Best score: ", clf.best_score_)

Best parameters:  {'C': 0.1, 'kernel': 'rbf'}
Best score:  0.6447031404580179
```

2. Logistic Regression-I:



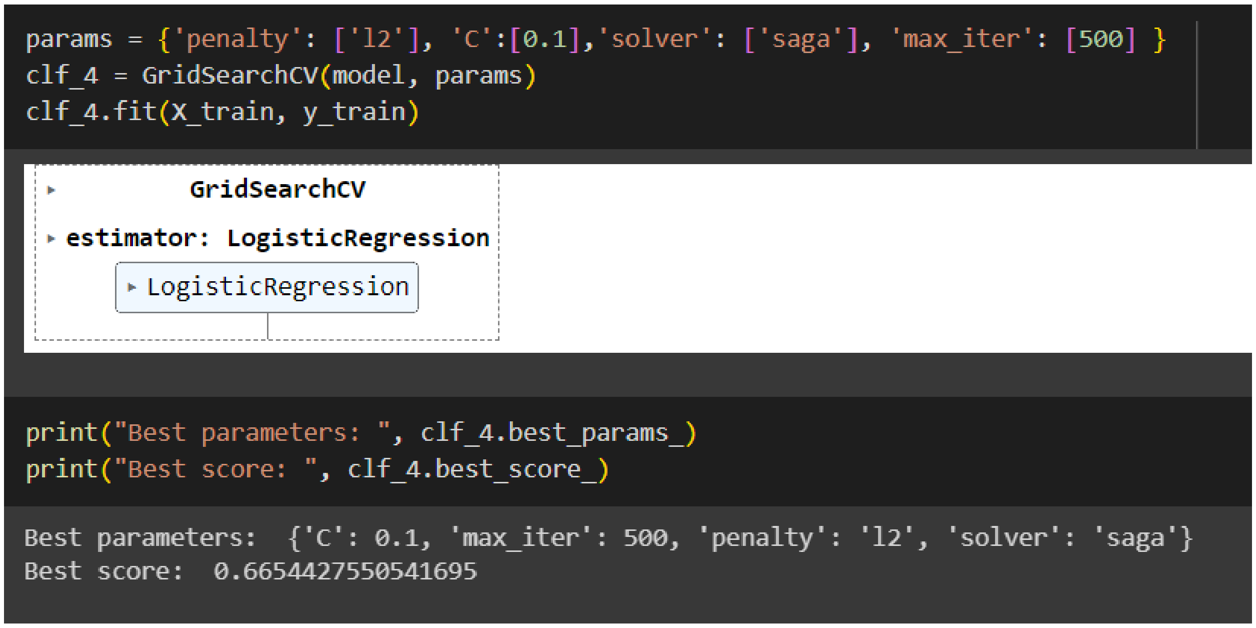
```
[ ] model = lr()
```

```
params = {'penalty': ['l2'], 'C':[0.1],'solver': ['liblinear'] }
clf_2 = GridSearchCV(model, params)
clf_2.fit(X_train, y_train)
```

```
[ ] print("Best parameters: ", clf_2.best_params_)
    print("Best score: ", clf_2.best_score_)
```

```
Best parameters:  {'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'}
Best score:  0.6674148329232917
```

3. Logistic Regression-II:

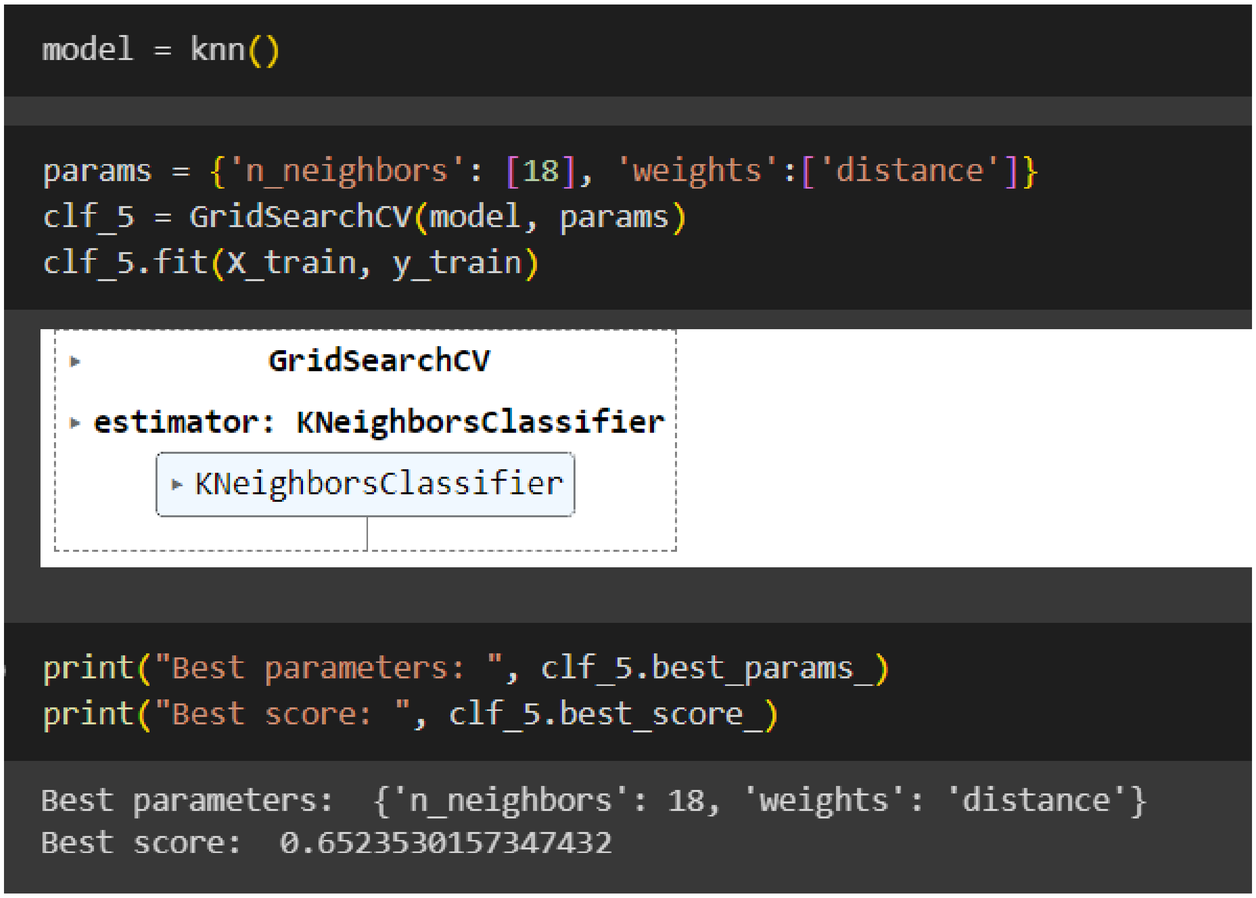


```
params = {'penalty': ['l2'], 'C':[0.1],'solver': ['saga'], 'max_iter': [500] }
clf_4 = GridSearchCV(model, params)
clf_4.fit(X_train, y_train)
```

```
print("Best parameters: ", clf_4.best_params_)
print("Best score: ", clf_4.best_score_)
```

```
Best parameters:  {'C': 0.1, 'max_iter': 500, 'penalty': 'l2', 'solver': 'saga'}
Best score:  0.6654427550541695
```

4. K Nearest Neighbours:



```
model = knn()
```

```
params = {'n_neighbors': [18], 'weights':['distance']}
clf_5 = GridSearchCV(model, params)
clf_5.fit(X_train, y_train)
```

```
▸ GridSearchCV
▸ estimator: KNeighborsClassifier
  ▸ KNeighborsClassifier
```

```
print("Best parameters: ", clf_5.best_params_)
print("Best score: ", clf_5.best_score_)
```

```
Best parameters:  {'n_neighbors': 18, 'weights': 'distance'}
Best score:  0.6523530157347432
```

Related Work:

We referred to these papers for our project.

1. **Predicting Effective Arguments with A Natural Language Processing[1]:**

   This paper aims to predict the argument classification of argumentative essays written by students in grades 6-12 using the DeBERTa model. It uses the concept of disentangled attention and enhanced mask decoder to target multiple features of the text. The data consists of two parts of training and testing, and the authors compare their model with other models such as BERT and RoBERTa, and show that their model outperforms them with the lowest cross entropy metric of 0.619.

2. **DeBERTa: Decoding-Enhanced BERT with Disentangled Attention[2]**:

   This paper proposes a novel model architecture DeBERTa that improves upon BERT and RoBERTa models using two novel techniques: a disentangled attention mechanism and an enhanced masked decoder. DeBERTa incorporates absolute word position embeddings right before the softmax layer where the model decodes the masked words. It also proposes a new virtual adversarial training algorithm termed Scale-invariant-Fine-Tuning (SiFT). DeBERTa is pre-trained on 78G of data from Wikipedia, BookCorpus, and OPENWEBTEXT and is scaled with 1.5 B parameters. It surpassed human performance on the SuperGLUE NLU benchmark.

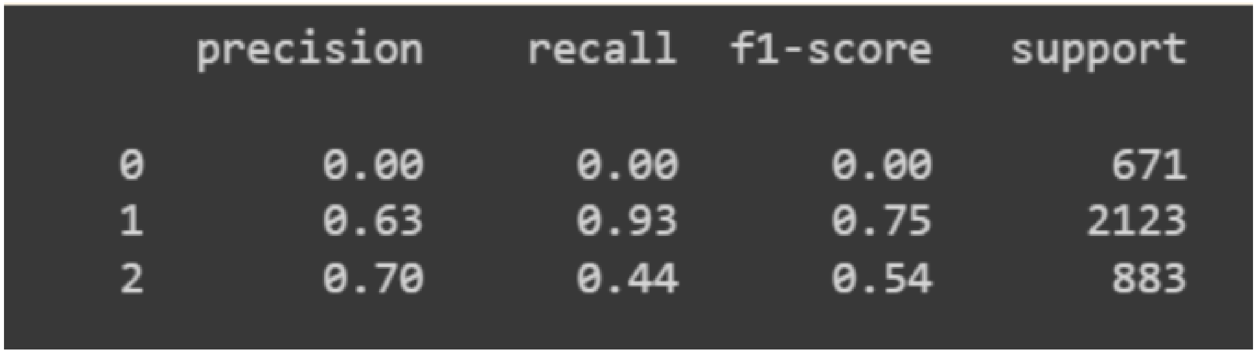
3. **Using Argument-based Features to Predict and Analyze Review Helpfulness[3]:**

   This paper investigates how argument-based features can be used to identify the usefulness of a review. The dataset used is a Tripadvisor hotel reviews corpus, where each review is broken down into 7 argument components. Fleiss' kappa metric was used to evaluate the quality of annotations. Four baseline features were considered: structural, Unigram, emotional, and semantic. The methods used achieved a performance enhancement, with an average improvement of 11.01% in terms of the F1-score and 10.40% in terms of AUC.

Results:

The following are the results generated by our models:

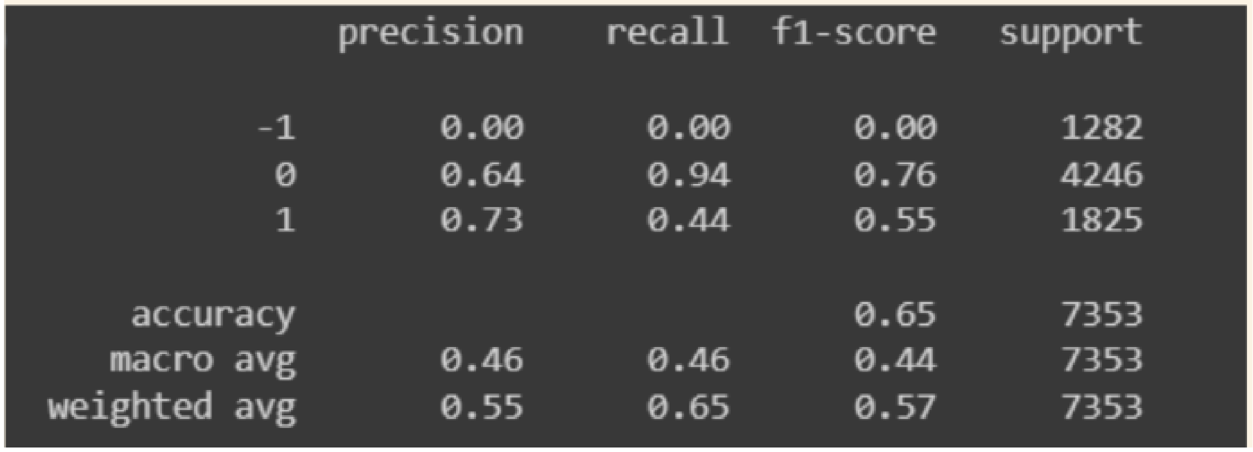
1. **BERT:**



```
   precision    recall  f1-score   support

0       0.00      0.00      0.00       671
1       0.63      0.93      0.75      2123
2       0.70      0.44      0.54       883
```

2. **BERT + SVC:**

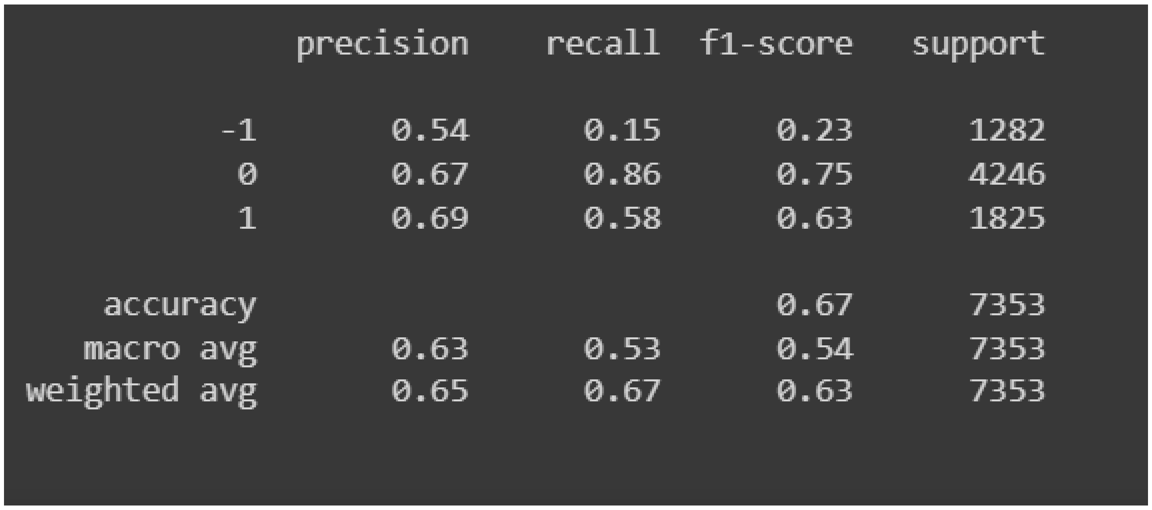


```
              precision    recall  f1-score   support

          -1       0.00      0.00      0.00      1282
           0       0.64      0.94      0.76      4246
           1       0.73      0.44      0.55      1825

    accuracy                           0.65      7353
   macro avg       0.46      0.46      0.44      7353
weighted avg       0.55      0.65      0.57      7353
```

3. **BERT + LR-I:**

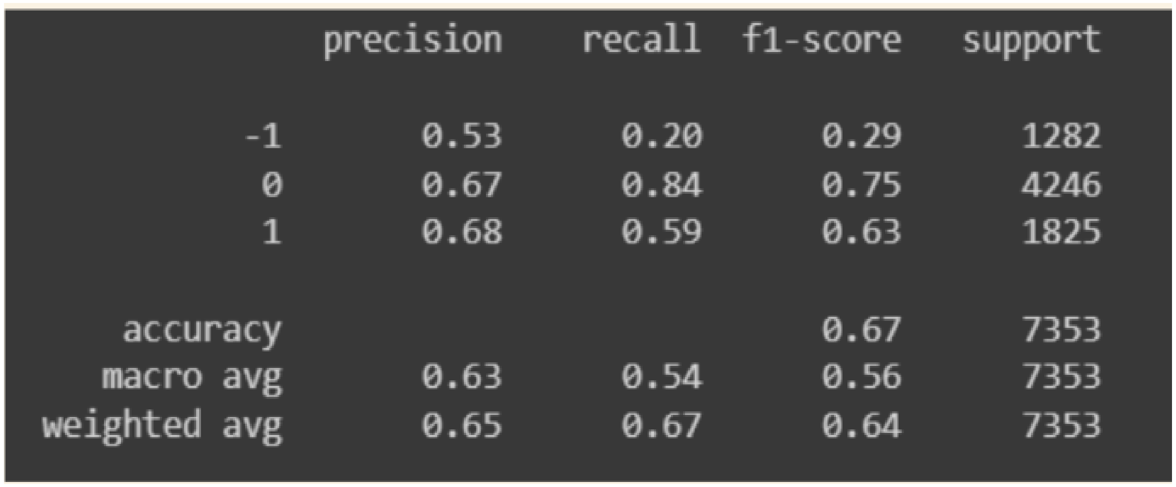


```
              precision    recall  f1-score   support

          -1       0.54      0.15      0.23      1282
           0       0.67      0.86      0.75      4246
           1       0.69      0.58      0.63      1825

    accuracy                           0.67      7353
   macro avg       0.63      0.53      0.54      7353
weighted avg       0.65      0.67      0.63      7353
```

4. **BERT + LR-II:**

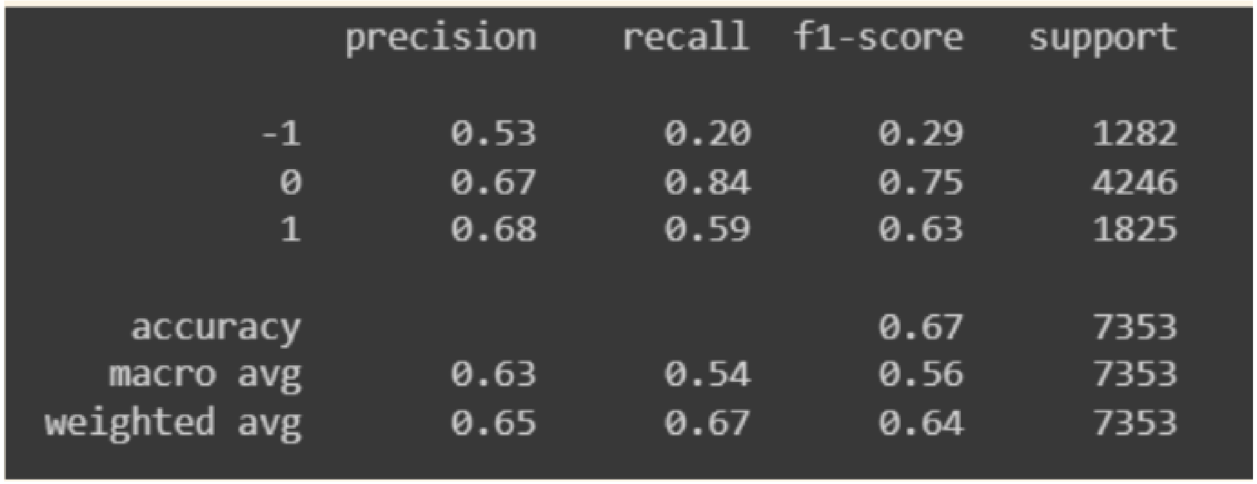


```
              precision    recall  f1-score   support

          -1       0.53      0.20      0.29      1282
           0       0.67      0.84      0.75      4246
           1       0.68      0.59      0.63      1825

    accuracy                           0.67      7353
   macro avg       0.63      0.54      0.56      7353
weighted avg       0.65      0.67      0.64      7353
```

5. **BERT + KNN:**



```
              precision    recall  f1-score   support

          -1       0.53      0.20      0.29      1282
           0       0.67      0.84      0.75      4246
           1       0.68      0.59      0.63      1825

    accuracy                           0.67      7353
   macro avg       0.63      0.54      0.56      7353
weighted avg       0.65      0.67      0.64      7353
```

Here as we can see the best results obtained are by using BERT and Logistic Regression, while using purely BERT provides a decent F-1 score for the classification of arguments as Effective and Highly Effective, it fails to classify arguments as Ineffective.

Challenges:

1. Bert-base and DeBERTa models are computationally time-consuming and need a lot of GPU RAM. While we were able to run the BERT-base model, we failed to run DeBERTa since we were limited by computational constraints.
2. Classifying arguments as Ineffective was very difficult, and the Recall was especially very poor for Ineffective arguments, our reasoning behind this issue is that, in real life it is very difficult for an argument to be completely ineffective, as there would be at least some discerning quality to each argument.

Conclusion:

We have successfully implemented a BERT-based model that is able to classify the quality of discourse elements in an argument. We have also found out that the best model for doing this task is a BERT + Logistic Regression model.

Future Scope:

1. We can look into fine-tuning DeBERTa or T5 language models for getting better results.
2. In order to overcome the challenge of less computational power and space available at hand, we can try to integrate the DeBERTa model with Amazon's AWS Service SageMaker, which provides a VCPU's, thus boosting computational power.
3. We may look into Ensemble Learning techniques wherein it is possible to combine several language models such as bert-base, DeBERTa and RoBERTa.

References:

1. https://ieeexplore.ieee.org/document/9950105
2. https://arxiv.org/pdf/2006.03654.pdf
3. http://aclanthology.lst.uni-saarland.de/D17-1142.pdf