eedi-submission-file

February 26, 2025

[ ]:
```
!pip install --no-deps --no-index /kaggle/input/hf-libraries/
 ↪sentence-transformers/sentence_transformers-3.1.0-py3-none-any.whl
```

```
Processing /kaggle/input/hf-libraries/sentence-
transformers/sentence_transformers-3.1.0-py3-none-any.whl
Installing collected packages: sentence-transformers
Successfully installed sentence-transformers-3.1.0
```

[ ]:
```
import pandas as pd
from sentence_transformers import SentenceTransformer, util
```

```
/opt/conda/lib/python3.10/site-
packages/sentence_transformers/cross_encoder/CrossEncoder.py:13:
TqdmExperimentalWarning: Using `tqdm.autonotebook.tqdm` in notebook mode. Use
`tqdm.tqdm` instead to force console mode (e.g. in jupyter console)
  from tqdm.autonotebook import tqdm, trange
```

[ ]:
```
import pandas as pd
test_df = pd.read_csv('/kaggle/input/eedi-mining-misconceptions-in-mathematics/
 ↪test.csv')
```

[ ]:
```
def map_correct_answer(row):
    if row['CorrectAnswer'] == 'A':
        return row['AnswerAText']
    elif row['CorrectAnswer'] == 'B':
        return row['AnswerBText']
    elif row['CorrectAnswer'] == 'C':
        return row['AnswerCText']
    elif row['CorrectAnswer'] == 'D':
        return row['AnswerDText']
```

[ ]:
```
test_df['Avalue'] = test_df.apply(map_correct_answer, axis=1)
```

[ ]:
```
test_df
```

[ ]:

	QuestionId	ConstructId	ConstructName	\
0	1869	856	Use the order of operations to carry out calcu…	
1	1870	1612	Simplify an algebraic fraction by factorising …	

```
2        1871         2774            Calculate the range from a list of data

   SubjectId                                          SubjectName CorrectAnswer  \
0         33                                              BIDMAS             A
1       1077                      Simplifying Algebraic Fractions             D
2        339  Range and Interquartile Range from a List of Data             B

                                        QuestionText            AnswerAText  \
0  \[\n3 \times 2+4-5\n\]\nWhere do the brackets …  \( 3 \times(2+4)-5 \)
1  Simplify the following, if possible: \( \frac{…              \( m+1 \)
2  Tom and Katie are discussing the \( 5 \) plant…              Only\nTom

              AnswerBText            AnswerCText             AnswerDText  \
0  \( 3 \times 2+(4-5) \)  \( 3 \times(2+4-5) \)  Does not need brackets
1              \( m+2 \)              \( m-1 \)       Does not simplify
2            Only\nKatie     Both Tom and Katie      Neither is correct

                  Avalue
0  \( 3 \times(2+4)-5 \)
1      Does not simplify
2            Only\nKatie
```

[ ]:
```python
def wide_to_long(df: pd.DataFrame) -> pd.DataFrame:
    # Fix the typo by adding a comma between "CorrectAnswer" and "Avalue"
    df = pd.melt(
        df[
            [
                "QuestionId",
                "QuestionText",
                "SubjectName",
                "ConstructName",
                "CorrectAnswer",  # Comma added here
                "Avalue",
                "AnswerAText",
                "AnswerBText",
                "AnswerCText",
                "AnswerDText"
            ]
        ],
        id_vars    = ["QuestionId", "QuestionText",␣
 ↪"CorrectAnswer","SubjectName", "Avalue", "ConstructName"],
        var_name   = 'Answer',
        value_name = 'value'
    )

    return df
```

```
test_long = wide_to_long(test_df)
```

[ ]:
```
test_long
```

[ ]:
```
    QuestionId                                       QuestionText  \
0         1869  \[\n3 \times 2+4-5\n\]\nWhere do the brackets …
1         1870  Simplify the following, if possible: \( \frac{…
2         1871  Tom and Katie are discussing the \( 5 \) plant…
3         1869  \[\n3 \times 2+4-5\n\]\nWhere do the brackets …
4         1870  Simplify the following, if possible: \( \frac{…
5         1871  Tom and Katie are discussing the \( 5 \) plant…
6         1869  \[\n3 \times 2+4-5\n\]\nWhere do the brackets …
7         1870  Simplify the following, if possible: \( \frac{…
8         1871  Tom and Katie are discussing the \( 5 \) plant…
9         1869  \[\n3 \times 2+4-5\n\]\nWhere do the brackets …
10        1870  Simplify the following, if possible: \( \frac{…
11        1871  Tom and Katie are discussing the \( 5 \) plant…

   CorrectAnswer                                        SubjectName  \
0              A                                             BIDMAS
1              D                        Simplifying Algebraic Fractions
2              B  Range and Interquartile Range from a List of Data
3              A                                             BIDMAS
4              D                        Simplifying Algebraic Fractions
5              B  Range and Interquartile Range from a List of Data
6              A                                             BIDMAS
7              D                        Simplifying Algebraic Fractions
8              B  Range and Interquartile Range from a List of Data
9              A                                             BIDMAS
10             D                        Simplifying Algebraic Fractions
11             B  Range and Interquartile Range from a List of Data

                  Avalue                                      ConstructName  \
0   \( 3 \times(2+4)-5 \)  Use the order of operations to carry out calcu…
1       Does not simplify  Simplify an algebraic fraction by factorising …
2             Only\nKatie           Calculate the range from a list of data
3   \( 3 \times(2+4)-5 \)  Use the order of operations to carry out calcu…
4       Does not simplify  Simplify an algebraic fraction by factorising …
5             Only\nKatie           Calculate the range from a list of data
6   \( 3 \times(2+4)-5 \)  Use the order of operations to carry out calcu…
7       Does not simplify  Simplify an algebraic fraction by factorising …
8             Only\nKatie           Calculate the range from a list of data
9   \( 3 \times(2+4)-5 \)  Use the order of operations to carry out calcu…
10      Does not simplify  Simplify an algebraic fraction by factorising …
11            Only\nKatie           Calculate the range from a list of data

         Answer                   value
```

```
0   AnswerAText   \( 3 \times(2+4)-5 \)
1   AnswerAText               \( m+1 \)
2   AnswerAText               Only\nTom
3   AnswerBText  \( 3 \times 2+(4-5) \)
4   AnswerBText               \( m+2 \)
5   AnswerBText             Only\nKatie
6   AnswerCText   \( 3 \times(2+4-5) \)
7   AnswerCText               \( m-1 \)
8   AnswerCText      Both Tom and Katie
9   AnswerDText  Does not need brackets
10  AnswerDText       Does not simplify
11  AnswerDText      Neither is correct
```

[ ]:
```
test_long = test_long.sort_values(["QuestionId", "Answer"]).
 ↪reset_index(drop=True)
test_long
```

[ ]:
```
    QuestionId                                       QuestionText  \
0         1869  \[\n3 \times 2+4-5\n\]\nWhere do the brackets …
1         1869  \[\n3 \times 2+4-5\n\]\nWhere do the brackets …
2         1869  \[\n3 \times 2+4-5\n\]\nWhere do the brackets …
3         1869  \[\n3 \times 2+4-5\n\]\nWhere do the brackets …
4         1870  Simplify the following, if possible: \( \frac{…
5         1870  Simplify the following, if possible: \( \frac{…
6         1870  Simplify the following, if possible: \( \frac{…
7         1870  Simplify the following, if possible: \( \frac{…
8         1871  Tom and Katie are discussing the \( 5 \) plant…
9         1871  Tom and Katie are discussing the \( 5 \) plant…
10        1871  Tom and Katie are discussing the \( 5 \) plant…
11        1871  Tom and Katie are discussing the \( 5 \) plant…

   CorrectAnswer                                      SubjectName  \
0              A                                           BIDMAS
1              A                                           BIDMAS
2              A                                           BIDMAS
3              A                                           BIDMAS
4              D                    Simplifying Algebraic Fractions
5              D                    Simplifying Algebraic Fractions
6              D                    Simplifying Algebraic Fractions
7              D                    Simplifying Algebraic Fractions
8              B  Range and Interquartile Range from a List of Data
9              B  Range and Interquartile Range from a List of Data
10             B  Range and Interquartile Range from a List of Data
11             B  Range and Interquartile Range from a List of Data

                  Avalue                                    ConstructName  \
0   \( 3 \times(2+4)-5 \)  Use the order of operations to carry out calcu…
```

```
1    \( 3 \times(2+4)-5 \)  Use the order of operations to carry out calcu…
2    \( 3 \times(2+4)-5 \)  Use the order of operations to carry out calcu…
3    \( 3 \times(2+4)-5 \)  Use the order of operations to carry out calcu…
4        Does not simplify  Simplify an algebraic fraction by factorising …
5        Does not simplify  Simplify an algebraic fraction by factorising …
6        Does not simplify  Simplify an algebraic fraction by factorising …
7        Does not simplify  Simplify an algebraic fraction by factorising …
8              Only\nKatie            Calculate the range from a list of data
9              Only\nKatie            Calculate the range from a list of data
10             Only\nKatie            Calculate the range from a list of data
11             Only\nKatie            Calculate the range from a list of data
```

	Answer	value
0	AnswerAText	\(3 \times(2+4)-5 \)
1	AnswerBText	\(3 \times 2+(4-5) \)
2	AnswerCText	\(3 \times(2+4-5) \)
3	AnswerDText	Does not need brackets
4	AnswerAText	\(m+1 \)
5	AnswerBText	\(m+2 \)
6	AnswerCText	\(m-1 \)
7	AnswerDText	Does not simplify
8	AnswerAText	Only\nTom
9	AnswerBText	Only\nKatie
10	AnswerCText	Both Tom and Katie
11	AnswerDText	Neither is correct

[ ]:
```
test_long["Answer_alphabet"] = test_long["Answer"].str.
 ↪extract(r'Answer([A-Z])Text$')
test_long["QuestionId_Answer"] = test_long["QuestionId"].astype("str") + "_" +␣
 ↪test_long["Answer_alphabet"]
test_long = test_long[test_long["CorrectAnswer"] !=␣
 ↪test_long["Answer_alphabet"]]
```

[ ]:
```
test_long.columns
```

[ ]:
```
Index(['QuestionId', 'QuestionText', 'CorrectAnswer', 'SubjectName', 'Avalue',
       'ConstructName', 'Answer', 'value', 'Answer_alphabet',
       'QuestionId_Answer'],
      dtype='object')
```

[ ]:
```
!pip install --no-deps --no-index /kaggle/input/hf-libraries/
 ↪sentence-transformers/sentence_transformers-3.1.0-py3-none-any.whl
```

```
/opt/conda/lib/python3.10/pty.py:89: RuntimeWarning: os.fork() was called.
os.fork() is incompatible with multithreaded code, and JAX is multithreaded, so
this will likely lead to a deadlock.
  pid, fd = os.forkpty()
```

```
Processing /kaggle/input/hf-libraries/sentence-
transformers/sentence_transformers-3.1.0-py3-none-any.whl
sentence-transformers is already installed with the same version as the provided
wheel. Use --force-reinstall to force an installation of the wheel.
```

[ ]:
```
model = SentenceTransformer('/kaggle/input/finetune_bge_large/pytorch/default/1/
↪trained_model')
```

[ ]:
```
model1 = SentenceTransformer('/kaggle/input/all_mpnet/pytorch/default/1/content/
↪trained_model')
```

[ ]:
```
def batch_encode_texts(texts, model, model1, batch_size=32):
    embeddings = []
    embeddings1 = []
    for i in range(0, len(texts), batch_size):
        batch = texts[i:i+batch_size]
        embeddings.append(model.encode(batch, convert_to_tensor=True))
        embeddings1.append(model1.encode(batch, convert_to_tensor=True))
    return embeddings, embeddings1
```

[ ]:
```

```

[ ]:
```
def precompute_embeddings(misconceptions, model):
    return model.encode(misconceptions, convert_to_tensor=True)
```

[ ]:
```
def precompute_embeddings1(misconceptions, model1):
    return model1.encode(misconceptions, convert_to_tensor=True)
```

[ ]:
```
def common_word_overlap(text1, text2):
    set1 = set(text1.lower().split())
    set2 = set(text2.lower().split())
    common_words = set1 & set2  # Intersection
    total_words = set1 | set2   # Union
    return len(common_words) / len(total_words) if total_words else 0
```

[ ]:
```
test_long = test_long.copy()
```

[ ]:
```
test_long.loc[:, 'complete_text'] = (
    test_long['ConstructName'] + ' ' +
    test_long['ConstructName'] + ' ' +
    test_long['QuestionText'] + ' ' +
    test_long['Avalue'] + ' ' +
    test_long['value']
)
```

[ ]:
```
text_list = test_long.complete_text.to_list()
```

```
[ ]: misconception_mapping = pd.read_csv('/kaggle/input/
     ↪eedi-mining-misconceptions-in-mathematics/misconception_mapping.csv')
```

```
[ ]: misconceptions = misconception_mapping.MisconceptionName.to_list()
```

```
[ ]: misconception_embeddings = precompute_embeddings(misconceptions, model)
```

```
Batches:   0%|          | 0/81 [00:00<?, ?it/s]
```

```
[ ]: misconception_embeddings.shape
```

```
[ ]: torch.Size([2587, 1024])
```

```
[ ]: misconception_embeddings1 = precompute_embeddings1(misconceptions, model1)
```

```
Batches:   0%|          | 0/81 [00:00<?, ?it/s]
```

```
[ ]: misconception_embeddings1.shape
```

```
[ ]: torch.Size([2587, 768])
```

```
[ ]: misconception_mapping.head(1)
```

```
[ ]:    MisconceptionId                                MisconceptionName
     0                0  Does not know that angles in a triangle sum to…
```

```
[ ]: def compute_all_similarities(text_batch, text_batch1, misconception_embeddings,␣
     ↪misconception_embeddings1, misconception_mapping, text_list, top_k=25):
         # Cosine similarities (efficient matrix multiplication)
         cosine_similarities = util.cos_sim(text_batch, misconception_embeddings)
         cosine_similarities1 = util.cos_sim(text_batch1, misconception_embeddings1)

         # List to store final scores for top misconceptions
         top_k_indices_list = []

         # Convert the text_batch tensors back to original text for common word␣
     ↪overlap calculation
         # Assuming the text_df is accessible globally or passed as a parameter
         original_texts = text_list  # List of original texts

         for i in range(len(text_batch)):
             # For each text, calculate semantic similarity using util.
     ↪semantic_search
             semantic_similarities = util.semantic_search(text_batch[i].
     ↪unsqueeze(0), misconception_embeddings, top_k=len(misconception_mapping))[0]
```

```
        semantic_similarities1 = util.semantic_search(text_batch1[i].
↪unsqueeze(0), misconception_embeddings1, top_k=len(misconception_mapping))[0]

        combined_scores = []

        # Iterate over misconceptions using their IDs and names
        for j, (misconception_id, misconception) in␣
↪enumerate(zip(misconception_mapping['MisconceptionId'],␣
↪misconception_mapping['MisconceptionName'])):
            # Common word overlap score
            # Access the original text directly from the list
            common_word_score = common_word_overlap(original_texts[i],␣
↪misconception)

            # Cosine similarity score
            cosine_similarity_score = cosine_similarities[i][j].item()
            cosine_similarity_score1 = cosine_similarities1[i][j].item()

            # Semantic similarity score (from semantic_search)
            semantic_similarity_score = semantic_similarities[j]['score']
            semantic_similarity_score1 = semantic_similarities1[j]['score']

            # Combine all three metrics
            combined_score = (0.04*common_word_score + 0.
↪42*cosine_similarity_score + 0.03*semantic_similarity_score + 0.
↪48*cosine_similarity_score1 + 0.03*semantic_similarity_score1) / 5

            # Store combined scores for this text-misconception pair
            combined_scores.append((misconception_id, combined_score))

        # Sort misconceptions by the combined score in descending order
        sorted_combined_scores = sorted(combined_scores, key=lambda x: x[1],␣
↪reverse=True)

        # Get the top_k misconceptions (top 25 by default)
        top_k_indices = [misconception_id for misconception_id, score in␣
↪sorted_combined_scores[:top_k]]

        top_k_indices_list.append(top_k_indices)

    return top_k_indices_list
```

[ ]:
```
batch_size = 8
text_embeddings_batches_m,text_embeddings_batches_m1 =␣
↪batch_encode_texts(test_long['complete_text'].tolist(), model, model1,␣
↪batch_size=batch_size)
```

```
Batches:   0%|          | 0/1 [00:00<?, ?it/s]
Batches:   0%|          | 0/1 [00:00<?, ?it/s]
Batches:   0%|          | 0/1 [00:00<?, ?it/s]
Batches:   0%|          | 0/1 [00:00<?, ?it/s]
```

[ ]:
```
text_embeddings_batches_m[0].shape
```

[ ]:
```
torch.Size([8, 1024])
```

[ ]:
```
text_embeddings_batches_m1[0].shape
```

[ ]:
```
torch.Size([8, 768])
```

[ ]:
```
text_list = test_long['complete_text'].tolist()
top_k_indices_list = []

# For each batch of text embeddings, compute the top K matching misconceptions
for text_batch,text_batch1 in zip(text_embeddings_batches_m,␣
 ↪text_embeddings_batches_m1):
    # Compute all similarities and get top K misconceptions
    batch_top_k_indices = compute_all_similarities(
            text_batch, text_batch1, misconception_embeddings,␣
 ↪misconception_embeddings1, misconception_mapping, text_list, top_k=25
    )
    top_k_indices_list.extend(batch_top_k_indices)
```

[ ]:
```
# Ensure test_long is a copy
test_long = test_long.copy()

# Now you can assign the values without triggering the warning
test_long.loc[:, "MisconceptionId"] = [" ".join([str(x) for x in␣
 ↪top_k_indices_list]) for top_k_indices_list in top_k_indices_list]
```

1 finetuning

[ ]:
```
import numpy as np
def apk(actual, predicted, k=25):
    if not actual:
        return 0.0

    if len(predicted)>k:
        predicted = predicted[:k]

    score = 0.0
    num_hits = 0.0
```

```
    for i,p in enumerate(predicted):
        # first condition checks whether it is valid prediction
        # second condition checks if prediction is not repeated
        if p in actual and p not in predicted[:i]:
            num_hits += 1.0
            score += num_hits / (i+1.0)

    return score / min(len(actual), k)


def mapk(actual, predicted, k=25):
    return np.mean([apk(a,p,k) for a,p in zip(actual, predicted)])
```

[ ]:
```
finetune_df = test_long.copy()
```

2 checking score

[ ]:
```
import numpy as np

def apk(actual, predicted, k=25):
    """ Computes the Average Precision at k (AP@k). """
    if not actual:
        return 0.0

    predicted = predicted[:k]
    score = 0.0
    num_hits = 0.0

    for i, p in enumerate(predicted):
        if p in actual and p not in predicted[:i]:
            num_hits += 1.0
            score += num_hits / (i + 1.0)

    return score / min(len(actual), k)

def mapk(actual, predicted, k=25):
    """ Computes the Mean Average Precision at k (MAP@k). """
    return np.mean([apk(a, p, k) for a, p in zip(actual, predicted)])
```

[ ]:
```
df_train = pd.read_csv('/kaggle/input/eedi-mining-misconceptions-in-mathematics/
↪train.csv')
```

[ ]:
```
df_train.fillna(-1, inplace=True)
```

[ ]:
```
from tqdm import tqdm
```

[ ]:
```
df_label = {}
for idx, row in tqdm(df_train.iterrows(), total=len(df_train)):
    for option in ["A", "B", "C", "D"]:
        if (row.CorrectAnswer!=option) & (row[f"Misconception{option}Id"]!=-1):
            #df[f"{row.QuestionId}_{option}"] = apply_template(row, tokenizer,␣
 ↪option)
            df_label[f"{row.QuestionId}_{option}"] =␣
 ↪[row[f"Misconception{option}Id"]]

df_label = pd.DataFrame([df_label]).T.reset_index()
df_label.columns = ["QuestionId_Answer", "MisconceptionId"]
```

```
100%|      | 1869/1869 [00:00<00:00, 6895.14it/s]
```

[ ]:
```
import pandas as pd
predicted = test_long["MisconceptionId"].apply(lambda x: [int(y) for y in x.
 ↪split()])
label = df_label["MisconceptionId"]
print("Validation: ", mapk(label, predicted))
```

```
Validation:  0.19444444444444445
```

[ ]:
```

```

[ ]:
```
submission = test_long[["QuestionId_Answer", "MisconceptionId"]].
 ↪reset_index(drop=True)
```

[ ]:
```
submission['MisconceptionId'].iloc[0]
```

[ ]:
```
'1672 2306 2532 1941 2586 328 2488 15 1085 1516 2140 2518 1054 466 1119 706 2181
203 102 77 1005 752 1856 1507 2512'
```

[ ]:
```
submission
```

[ ]:

	QuestionId_Answer	MisconceptionId
0	1869_B	1672 2306 2532 1941 2586 328 2488 15 1085 1516…
1	1869_C	1672 2306 2532 1941 328 2488 2586 2140 1516 15…
2	1869_D	2306 1672 1941 2488 15 2586 2140 1005 1516 328…
3	1870_A	167 113 2068 2142 353 1540 3 265 46 633 79 606…
4	1870_B	167 113 353 2068 2142 1540 3 265 46 79 633 525…
5	1870_C	167 113 2068 2142 353 1540 3 265 46 1432 79 52…
6	1871_A	1287 1073 365 1923 1797 2151 2439 2319 129 247…
7	1871_C	1287 1073 365 1923 1797 2151 2439 2319 129 247…
8	1871_D	1287 1073 365 2151 1923 1797 2439 2319 129 247…

[ ]:
```
import os
```

```
if os.path.exists("/kaggle/working/submission.csv"):
    os.remove("/kaggle/working/submission.csv")

submission.to_csv("/kaggle/working/submission.csv", index=False)
```