```
# !pip install matplotlib pandas transformers seaborn
```

Проект по анализу данных "Зависимость результатов multiple-choice классификации от семантической близости классов"

Гипотеза

H0: Семантическая близость классов обратно пропорциональна precision LLM классификатора.

H1: Семантическая близость классов пропорциональна precision LLM классификатора.

Данные

Данные взяты из бенчмарка дискурсивных явлений DISRP (русскоязычный сабсет). Перед моделью стоит задача классификации дискурсивных отношений между предложениями (частями предложений). Всего в датасете 22 класса. Для сравнения семантической близости классов использовался bge-small-en-v1.5.

Методология

Для проверки гипотезы мы нашли для каждого класса в датасете DISRP 4 ближайших класс по семантической близости и 4 самых дальних класса. Таким образом мы собрали 4 варианта датасета, различающихся только набором классов подаваемых модели на вход для выбора: 5 ближайших классов, 5 дальних классов, 5 случайных классов (таргетный включен) и 22 класса (все классы). Мы проскорили модель gpt-4o-mini на каждом из этих датасетов и сравнили precision для каждого из них, а также посчитали корреляцию между precision и семантической близостью классов внутри сабсета.

```
import json
import random
import numpy as np
import pandas as pd
from tqdm import tqdm
from transformers import pipeline
from sklearn.cluster import KMeans
from sklearn.manifold import TSNE
from sklearn.preprocessing import normalize
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics import (
    accuracy_score,
    precision_recall_fscore_support,
    confusion_matrix,
```

```
    classification_report,
)
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import pearsonr
```

Data loading

```
with open("full_disrpt.json", "r", encoding="utf-8") as f:
    data = json.load(f)

# embedder = pipeline("feature-extraction", model="DeepPavlov/rubert-base-cased")
# embedder = pipeline("feature-extraction", model="cointegrated/LaBSE-en-ru")

embedder = pipeline("feature-extraction", model="BAAI/bge-small-en-v1.5")


def get_embedding(text):
    """
    Get the embedding for a given text using the embedder pipeline.
    """
    embedding = embedder(text)
    return np.mean(embedding[0], axis=0)
```

```
Device set to use cuda:0
```

```
df = pd.DataFrame(columns=["start", "end", "label"])
for item in data.values():
    row = {
        # '_id': item['id'],
        "start": item["sent_1"],
        "end": item["sent_2"],
        "label": item["label"],
    }
    df.loc[-1] = row
    df.index = df.index + 1
df.head()
```

```
                                                  start  \
28867  - формируются коэффициенты модулей начальной и...
28866  В этой статье решено привести обобщение алгори...
28865                                Кабардинец везёт хлеб
28864  ##### — Какие именно из всего семейства лиценз...
28863  Помимо правильного рабочего места , о котором ...

                                                    end          label
```

```
28867                                          [ формула ] ;   elaboration

28866  которые могут быть описаны одной или нескольки...   elaboration

28865                                - едет в Подольск ,         joint

28864  ##### — Проекты фонда « Викимедиа » , лицензир...  solutionhood

28863  Регулярное выполнение асан поможет сохранить з...   elaboration
```

```
df.info()
```

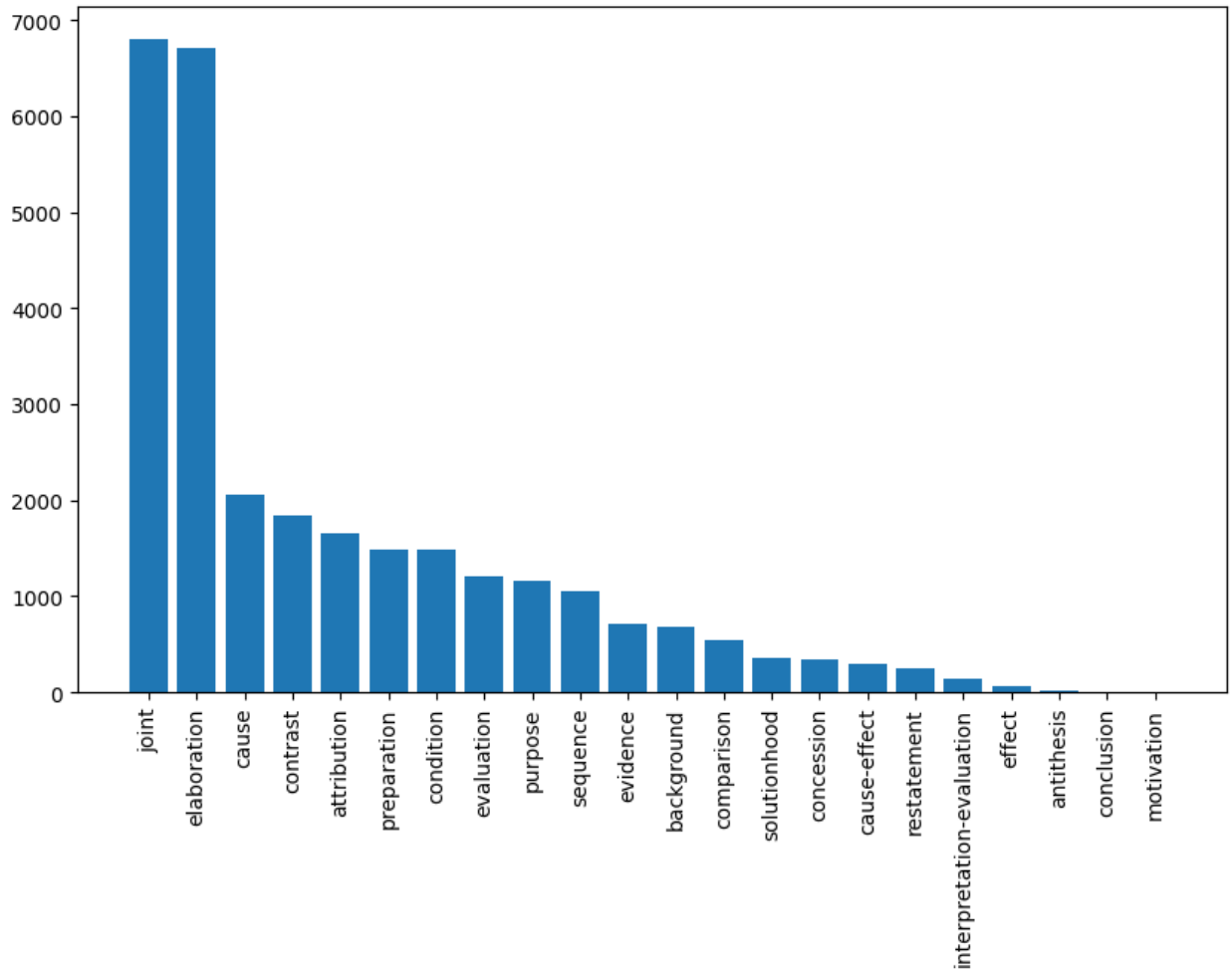
```
<class 'pandas.core.frame.DataFrame'>
Index: 28868 entries, 28867 to 0
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   start   28868 non-null  object
 1   end     28868 non-null  object
 2   label   28868 non-null  object
dtypes: object(3)
memory usage: 902.1+ KB
```

```
labels = df["label"].unique()
print(f"{len(labels)} Labels: {labels}")
```

```
22 Labels: ['elaboration' 'joint' 'solutionhood' 'cause' 'contrast'
'sequence'
 'purpose' 'preparation' 'concession' 'restatement' 'attribution'
 'evidence' 'evaluation' 'condition' 'background' 'cause-effect'
 'comparison' 'interpretation-evaluation' 'effect' 'antithesis'
 'conclusion' 'motivation']
```

```
plt.figure(figsize=(10, 6))
labels_amount = df["label"].value_counts()
plt.xticks(rotation=90)
plt.bar(labels_amount.index, labels_amount.values)
```

```
<BarContainer object of 22 artists>
```



Как можно заметить, дисбаланс классов в датасете достаточно большой, однако, для наших целей это не критично, т.к. мы не будем ничего обучать, а просто посмотрим на то, как ведут себя модели с различной комбинацией вариантов ответа. Всё же, нам нужно будет выделить сабсет учитывающий дисбаланс классов.

```python
# Create a test set for scoring
test_set = df.sample(frac=0.05, random_state=1)
print(f"Test set size: {len(test_set)}")
print(f"{len(test_set['label'].unique())} labels in the test set")
print(test_set["label"].value_counts())
```

```
Test set size: 1443
21 labels in the test set
label
joint                          332
elaboration                    317
contrast                       107
cause                           99
attribution                     90
condition                       76
```

```
preparation                  65
evaluation                   59
purpose                      57
sequence                     54
background                   45
evidence                     38
comparison                   29
concession                   18
solutionhood                 14
interpretation-evaluation    12
cause-effect                 11
restatement                  11
antithesis                    5
effect                        3
motivation                    1
Name: count, dtype: int64
```

```
test_set.head()
```

```
                                                    start  \
28831              Дочь вышла замуж за некоего Гуарама ,
14161  1 ) разной степени специализации фонемных приз...
25900                  чтобы выбрать наиболее эффективный
21208  " Ты , и только ты единственный/единственная ,...
15591  Нельзя сказать , что реальным сеньорам известн...

                                                      end       label
28831  Их первенец Баграт дал имя одному из самых изв...       cause
14161  3 ) разной вариативности и количества артикуля...       joint
25900                          при разных трудозатратах .   condition
21208  Да , в обмен на эту связь и ощущение безопасно...  evaluation
15591  Они , скорее , отличаются особой структурой зн...    contrast
```

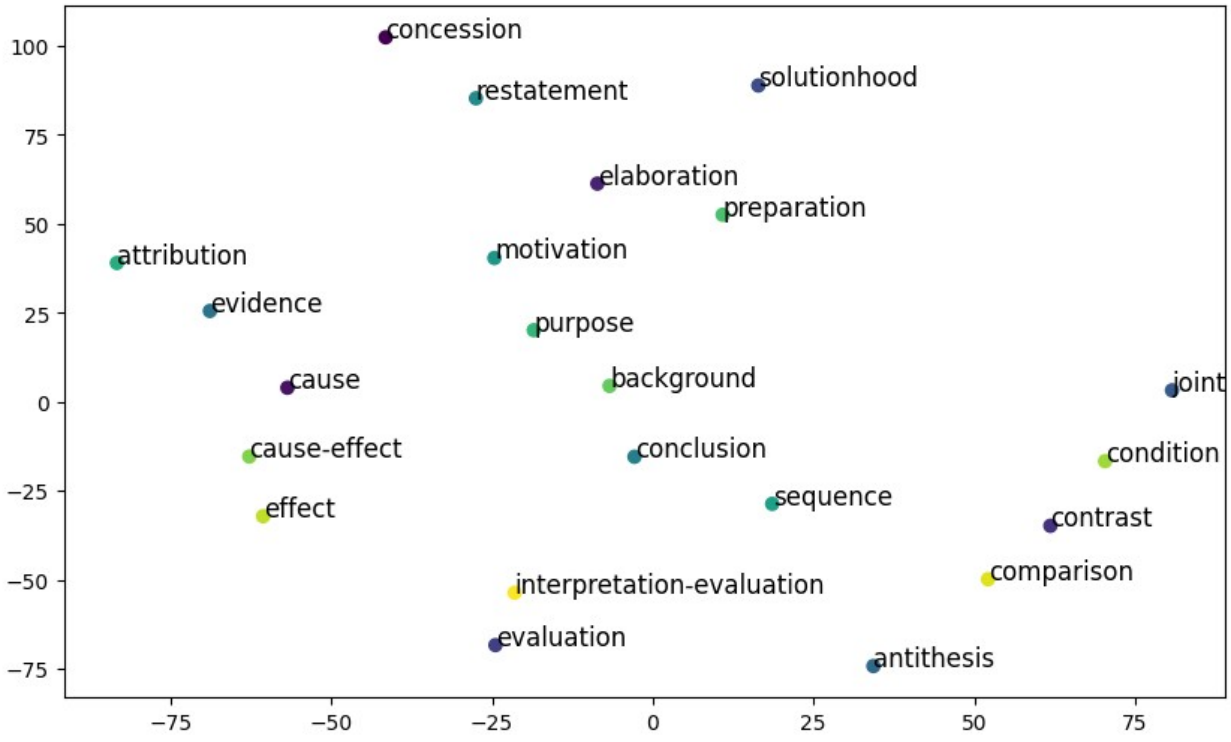
Similar or contrasting labels

Теперь мы можем попробовать найти для каждого класса наиболее и наименее похожие на него классы. Для этого мы используем сравнение векторов классов -- мы можем сравнить эмбеддинги названий классов и найти наиболее похожие и непохожие классы по косинусной близости.

```
label_embeddings = []
for label in labels:
    embedding = get_embedding(label)
    label_embeddings.append({"label": label, "embedding": embedding})

labels_emb_df = pd.DataFrame(label_embeddings)
labels_emb_df.to_csv("label_embeddings_rubert.csv", index=False)
```

```
# Clustering labels
label_embeddings = np.array(labels_emb_df["embedding"].tolist())
label_embeddings = normalize(label_embeddings)

k = len(labels)  # Number of clusters equal to number of unique labels
kmeans = KMeans(n_clusters=k, random_state=42).fit(label_embeddings)

# Reduce dimensions for visualization
tsne = TSNE(n_components=2, random_state=42, perplexity=5)
label_embeddings = tsne.fit_transform(label_embeddings)

# Plotting the clusters with names
plt.figure(figsize=(10, 6))
for i, label in enumerate(labels):
    plt.annotate(label, (label_embeddings[i, 0], label_embeddings[i,
1]), fontsize=12)
plt.scatter(
    label_embeddings[:, 0],
    label_embeddings[:, 1],
    c=kmeans.labels_,
    cmap="viridis",
    marker="o",
)
```

```
<matplotlib.collections.PathCollection at 0x7cd50ce6f0b0>
```



```python
# Get top 4 closest labels to each label and top 4 furthest labels by cosine similarity

labels_dictionaty = {}
cos_sim_matrix = cosine_similarity(label_embeddings, label_embeddings)
for i, label in enumerate(labels):
    # Exclude the label itself by setting its similarity to -inf/+inf
    sim_scores = cos_sim_matrix[i].copy()
    sim_scores[i] = -np.inf  # for closest
    closest_idx = np.argsort(sim_scores)[-4:][::-1]
    sim_scores[i] = np.inf  # for furthest
    furthest_idx = np.argsort(sim_scores)[:4]
    labels_dictionaty[label] = {
        "closest": labels[closest_idx].tolist(),
        "furthest": labels[furthest_idx].tolist(),
    }
for label, values in labels_dictionaty.items():
    print(f"Label: {label}")
    print(f"  Closest: {values['closest']}")
    print(f"  Furthest: {values['furthest']}")
```

```
Label: elaboration
  Closest: ['restatement', 'concession', 'solutionhood',
'preparation']
  Furthest: ['antithesis', 'conclusion', 'sequence', 'evaluation']
Label: joint
  Closest: ['condition', 'contrast', 'comparison', 'sequence']
  Furthest: ['cause', 'cause-effect', 'evidence', 'effect']
Label: solutionhood
  Closest: ['preparation', 'elaboration', 'restatement', 'concession']
  Furthest: ['conclusion', 'evaluation', 'interpretation-evaluation',
'antithesis']
Label: cause
  Closest: ['evidence', 'cause-effect', 'attribution', 'background']
  Furthest: ['joint', 'condition', 'contrast', 'comparison']
Label: contrast
  Closest: ['comparison', 'condition', 'sequence', 'joint']
  Furthest: ['attribution', 'background', 'evidence', 'purpose']
Label: sequence
  Closest: ['antithesis', 'comparison', 'contrast', 'conclusion']
  Furthest: ['motivation', 'purpose', 'concession', 'restatement']
Label: purpose
  Closest: ['motivation', 'background', 'concession', 'attribution']
  Furthest: ['comparison', 'sequence', 'antithesis', 'contrast']
Label: preparation
  Closest: ['solutionhood', 'elaboration', 'restatement',
'concession']
  Furthest: ['conclusion', 'evaluation', 'interpretation-evaluation',
'antithesis']
Label: concession
```

```
  Closest: ['restatement', 'motivation', 'elaboration', 'purpose']
  Furthest: ['antithesis', 'sequence', 'comparison', 'conclusion']
Label: restatement
  Closest: ['concession', 'elaboration', 'motivation', 'purpose']
  Furthest: ['antithesis', 'sequence', 'conclusion', 'comparison']
Label: attribution
  Closest: ['evidence', 'background', 'cause', 'purpose']
  Furthest: ['contrast', 'condition', 'comparison', 'joint']
Label: evidence
  Closest: ['attribution', 'background', 'cause', 'purpose']
  Furthest: ['condition', 'contrast', 'joint', 'comparison']
Label: evaluation
  Closest: ['interpretation-evaluation', 'conclusion', 'effect',
'antithesis']
  Furthest: ['preparation', 'solutionhood', 'elaboration',
'restatement']
Label: condition
  Closest: ['joint', 'contrast', 'comparison', 'sequence']
  Furthest: ['evidence', 'cause', 'attribution', 'background']
Label: background
  Closest: ['attribution', 'evidence', 'purpose', 'motivation']
  Furthest: ['contrast', 'comparison', 'condition', 'sequence']
Label: cause-effect
  Closest: ['effect', 'cause', 'evidence', 'attribution']
  Furthest: ['joint', 'condition', 'contrast', 'comparison']
Label: comparison
  Closest: ['sequence', 'contrast', 'antithesis', 'condition']
  Furthest: ['purpose', 'background', 'motivation', 'attribution']
Label: interpretation-evaluation
  Closest: ['evaluation', 'conclusion', 'effect', 'antithesis']
  Furthest: ['preparation', 'solutionhood', 'elaboration',
'restatement']
Label: effect
  Closest: ['cause-effect', 'cause', 'interpretation-evaluation',
'evaluation']
  Furthest: ['joint', 'condition', 'preparation', 'solutionhood']
Label: antithesis
  Closest: ['sequence', 'comparison', 'conclusion', 'contrast']
  Furthest: ['concession', 'motivation', 'restatement', 'elaboration']
Label: conclusion
  Closest: ['evaluation', 'interpretation-evaluation', 'antithesis',
'sequence']
  Furthest: ['solutionhood', 'preparation', 'elaboration',
'restatement']
Label: motivation
  Closest: ['concession', 'purpose', 'restatement', 'elaboration']
  Furthest: ['sequence', 'antithesis', 'comparison', 'contrast']
```

Добавим в датасет столбцы с наиболее похожими и непохожими классами.

```
test_set["closest"] = test_set["label"].map(lambda x:
labels_dictionaty[x]["closest"])
test_set["furthest"] = test_set["label"].map(lambda x:
labels_dictionaty[x]["furthest"])
test_set["random"] = test_set["label"].map(
    lambda x: np.random.choice(
        test_set["label"].unique().tolist(), 4, replace=False
    ).tolist()
)
test_set["all"] = test_set["label"].map(lambda x:
test_set["label"].unique().tolist())
test_set.head()
```

```
                                                   start  \
28831              Дочь вышла замуж за некоего Гуарама ,
14161  1 ) разной степени специализации фонемных приз...
25900                 чтобы выбрать наиболее эффективный
21208  " Ты , и только ты единственный/единственная ,...
15591  Нельзя сказать , что реальным сеньорам известн...

                                                     end
label  \
28831  Их первенец Баграт дал имя одному из самых изв...       cause

14161  3 ) разной вариативности и количества артикуля...       joint

25900                          при разных трудозатратах .   condition

21208  Да , в обмен на эту связь и ощущение безопасно...  evaluation

15591  Они , скорее , отличаются особой структурой зн...    contrast


                                                  closest  \
28831  [evidence, cause-effect, attribution, background]
14161        [condition, contrast, comparison, sequence]
25900            [joint, contrast, comparison, sequence]
21208  [interpretation-evaluation, conclusion, effect...
15591           [comparison, condition, sequence, joint]

                                                 furthest  \
28831           [joint, condition, contrast, comparison]
14161            [cause, cause-effect, evidence, effect]
25900         [evidence, cause, attribution, background]
21208  [preparation, solutionhood, elaboration, resta...
15591       [attribution, background, evidence, purpose]

                                                   random  \
28831  [interpretation-evaluation, background, restat...
14161  [elaboration, attribution, concession, motivat...
```

```
25900     [attribution, restatement, preparation, joint]
21208    [condition, motivation, evidence, solutionhood]
15591  [evaluation, antithesis, background, interpret...
```

```
                                                     all
28831  [cause, joint, condition, evaluation, contrast...
14161  [cause, joint, condition, evaluation, contrast...
25900  [cause, joint, condition, evaluation, contrast...
21208  [cause, joint, condition, evaluation, contrast...
15591  [cause, joint, condition, evaluation, contrast...
```

```
label_to_emb = {row['label']: np.array(row['embedding']) for _, row in
labels_emb_df.iterrows()}

def avg_cosine_sim(col_name):
    sims = []
    for _, row in test_set.iterrows():
        label_emb = label_to_emb[row['label']]
        # Calculate average embedding for the choices
        choices_emb = np.mean([label_to_emb[choice] for choice in
row[col_name]], axis=0)

        sim = cosine_similarity([label_emb], [choices_emb])[0][0]
        sims.append(sim)
    return np.mean(sims)

similarities = {}

for col in ["closest", "furthest", "random", "all"]:
    avg_sim = avg_cosine_sim(col)
    similarities[col] = avg_sim
    # print(f"Average cosine similarity for '{col}': {avg_sim:.4f}")
similarities_df = pd.DataFrame.from_dict(similarities, orient='index',
columns=['Average Cosine Similarity'])
similarities_df.head()
```

```
          Average Cosine Similarity
closest                    0.716062
furthest                   0.645864
random                     0.706171
all                        0.759086
```

Scoring

for this part to run you need to have `.env` file with `OPENAI_API_KEY` and `OPENAI_BASE_URL` set.

```
from score_model import score_disrpt
```

```python
# Посчитаем метрики для тестового набора с разными вариантами
вариантов ответа
closest_set = test_set[["start", "end", "label", "closest"]].copy()
closest_set.rename(columns={"closest": "choices"}, inplace=True)
score_disrpt(
    model_name="gpt-4o-mini",
    temperature=0.5,
    test_set=closest_set,
    filename="gpt4o_mini_closest.csv",
)
```

```
[INFO] Starting disrpt scoring...

Scoring disrpt: 100%|██████████| 1443/1443 [15:02<00:00,  1.60it/s]

[INFO] Disrpt scoring complete. Results saved to:
gpt4o_mini_closest.csv


/home/askatasuna/Документы/HSE/contexts/ContextBench_2025/.venv/lib/
python3.12/site-packages/sklearn/metrics/_classification.py:1565:
UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in
labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))

({'accuracy': 0.46084546084546085,
  'recall': 0.36778899134802506,
  'f1': 0.2978047815372343,
  'precision': 0.3438929547918048},
 'gpt4o_mini_closest.csv')
```

```python
furthest_set = test_set[["start", "end", "label", "furthest"]].copy()
furthest_set.rename(columns={"furthest": "choices"}, inplace=True)
score_disrpt(
    model_name="gpt-4o-mini",
    temperature=0.5,
    test_set=furthest_set,
    filename="gpt4o_mini_furthest.csv",
)
```

```
[INFO] Starting disrpt scoring...

Scoring disrpt: 100%|██████████| 1443/1443 [15:19<00:00,  1.57it/s]

[INFO] Disrpt scoring complete. Results saved to:
gpt4o_mini_furthest.csv


/home/askatasuna/Документы/HSE/contexts/ContextBench_2025/.venv/lib/
python3.12/site-packages/sklearn/metrics/_classification.py:1565:
```

```
UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in
labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))

({'accuracy': 0.5904365904365905,
  'recall': 0.5801491793721999,
  'f1': 0.5071356889718956,
  'precision': 0.534008908567364},
 'gpt4o_mini_furthest.csv')
```

```
random_set = test_set[["start", "end", "label", "random"]].copy()
random_set.rename(columns={"random": "choices"}, inplace=True)
score_disrpt(
    model_name="gpt-4o-mini",
    temperature=0.5,
    test_set=random_set,
    filename="gpt4o_mini_random.csv",
)
```

```
[INFO] Starting disrpt scoring...

Scoring disrpt: 100%|██████████| 1443/1443 [14:15<00:00,  1.69it/s]

[INFO] Disrpt scoring complete. Results saved to:
gpt4o_mini_random.csv



({'accuracy': 0.483021483021483,
  'recall': 0.45412073944699666,
  'f1': 0.36052196556609817,
  'precision': 0.37058395331095284},
 'gpt4o_mini_random.csv')
```

```
all_set = test_set[["start", "end", "label", "all"]].copy()
all_set.rename(columns={"all": "choices"}, inplace=True)
score_disrpt(
    model_name="gpt-4o-mini",
    temperature=0.5,
    test_set=all_set,
    filename="gpt4o_mini_all.csv",
)
```

```
[INFO] Starting disrpt scoring...

Scoring disrpt: 100%|██████████| 1443/1443 [12:39<00:00,  1.90it/s]

[INFO] Disrpt scoring complete. Results saved to: gpt4o_mini_all.csv
```

```
/home/askatasuna/Документы/HSE/contexts/ContextBench_2025/.venv/lib/
python3.12/site-packages/sklearn/metrics/_classification.py:1565:
UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in
labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
/home/askatasuna/Документы/HSE/contexts/ContextBench_2025/.venv/lib/
python3.12/site-packages/sklearn/metrics/_classification.py:1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))

({'accuracy': 0.2945252945252945,
  'recall': 0.25364636118067724,
  'f1': 0.2166693633584598,
  'precision': 0.26670081557635805},
 'gpt4o_mini_all.csv')
```

Analyzing results

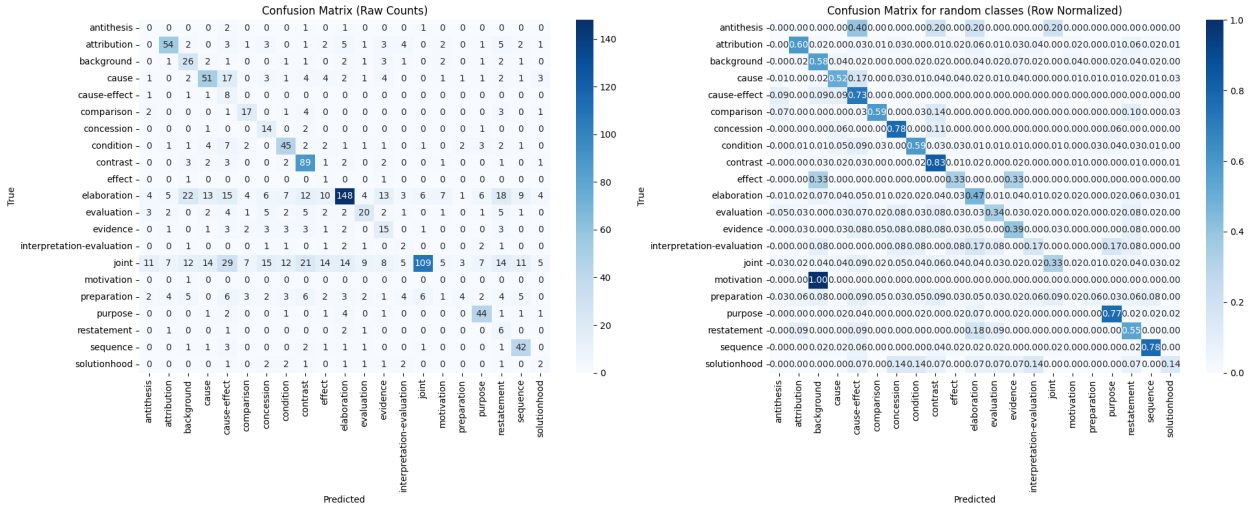
Теперь мы можем проанализировать результаты, которые мы получили. В первую очередь, мы можем посмотреть на confusion matrix, чтобы понять, какие классы чаще всего путаются между собой. А также мы можем посмотреть на корреляцию accuracy и сабсета.

```
similar_labels_df = pd.read_csv("gpt4o_mini_closest.csv")
different_labels_df = pd.read_csv("gpt4o_mini_furthest.csv")
random_labels_df = pd.read_csv("gpt4o_mini_random.csv")
all_labels_df = pd.read_csv("gpt4o_mini_all.csv")
```

Случайный набор классов

```
y_true = random_labels_df["correct_answer"].values
y_pred = random_labels_df["answer"].values

# Calculate accuracy
accuracy = accuracy_score(y_true, y_pred)
print(f"Accuracy: {accuracy:.4f}")

# Get unique labels (relations)
labels = sorted(list(set(y_true) | set(y_pred)))

precision, recall, f1, _ = precision_recall_fscore_support(
    y_true, y_pred, average="macro", labels=labels
)
print(f"Macro Precision: {precision:.4f}")
```

```
print(f"Macro Recall: {recall:.4f}")
print(f"Macro F1: {f1:.4f}")

# Create figure with two subplots side by side
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 8))

# 1. Regular confusion matrix
cm = confusion_matrix(y_true, y_pred, labels=labels)
sns.heatmap(
    cm,
    annot=True,
    fmt="d",
    cmap="Blues",
    xticklabels=labels,
    yticklabels=labels,
    ax=ax1,
)
ax1.set_xlabel("Predicted")
ax1.set_ylabel("True")
ax1.set_title("Confusion Matrix (Raw Counts)")

# 2. Row-normalized confusion matrix (normalize by true label)
cm_normalized = confusion_matrix(y_true, y_pred, labels=labels,
normalize="true")
sns.heatmap(
    cm_normalized,
    annot=True,
    fmt=".2f",
    cmap="Blues",
    xticklabels=labels,
    yticklabels=labels,
    ax=ax2,
)
ax2.set_xlabel("Predicted")
ax2.set_ylabel("True")
ax2.set_title("Confusion Matrix for random classes (Row Normalized)")

plt.tight_layout()
plt.savefig("confusion_matrices.png", dpi=300)
plt.show()

Accuracy: 0.4830
Macro Precision: 0.3706
Macro Recall: 0.4541
Macro F1: 0.3605
```




```
cm_df = pd.DataFrame(cm_normalized, index=labels, columns=labels)

misclassifications = []

for true_class in labels:
    for pred_class in labels:
        if true_class != pred_class:
            raw_count = cm[labels.index(true_class),
labels.index(pred_class)]

            # Only include pairs that actually appeared in the data
            if raw_count > 0:
                misclassifications.append(
                    {
                        "True Class": true_class,
                        "Predicted Class": pred_class,
                        "Error Rate": cm_df.loc[true_class,
pred_class],
                        "Count": raw_count,
                    }
                )

misclass_df = pd.DataFrame(misclassifications)
misclass_df = misclass_df.sort_values(by="Error Rate",
ascending=False)

print("Top confused pairs (by error rate):")
print(misclass_df.head(10))

misclass_df_by_count = misclass_df.sort_values(by="Count",
ascending=False)
print("\nTop confused pairs (by raw count):")
print(misclass_df_by_count.head(10))
```

```
Top confused pairs (by error rate):
                   True Class Predicted Class  Error Rate  Count
162                motivation      background    1.000000      1
0                  antithesis    cause-effect    0.400000      2
85                     effect      background    0.333333      1
86                     effect        evidence    0.333333      1
2                  antithesis     elaboration    0.200000      1
1                  antithesis        contrast    0.200000      1
3                  antithesis           joint    0.200000      1
192               restatement     elaboration    0.181818      2
34                      cause    cause-effect    0.171717     17
138  interpretation-evaluation     elaboration    0.166667      2

Top confused pairs (by raw count):
      True Class Predicted Class  Error Rate  Count
146        joint    cause-effect    0.087349     29
89   elaboration      background    0.069401     22
150        joint        contrast    0.063253     21
104  elaboration     restatement    0.056782     18
34         cause    cause-effect    0.171717     17
91   elaboration    cause-effect    0.047319     15
148        joint      concession    0.045181     15
151        joint          effect    0.042169     14
152        joint     elaboration    0.042169     14
159        joint     restatement    0.042169     14
```

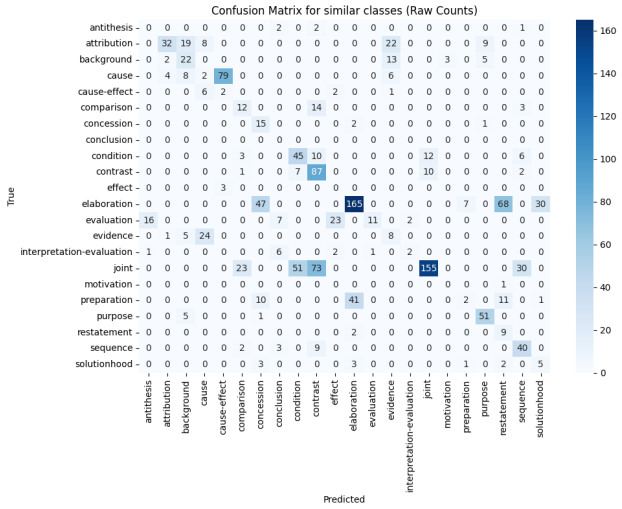
Наиболее похожие классы

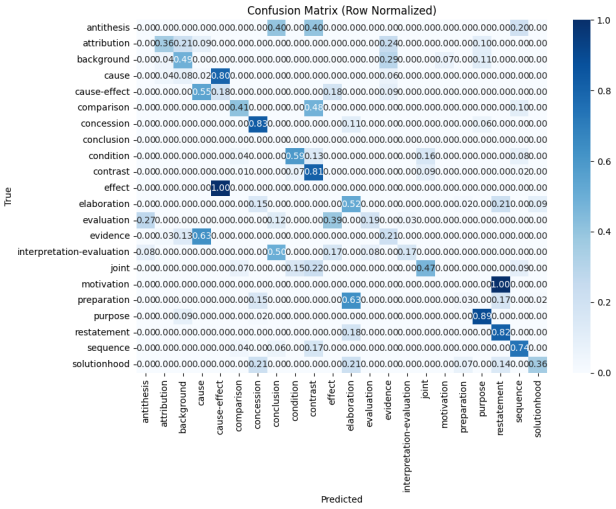
```python
y_true = similar_labels_df["correct_answer"].values
y_pred = similar_labels_df["answer"].values

# Calculate accuracy
accuracy = accuracy_score(y_true, y_pred)
print(f"Accuracy: {accuracy:.4f}")

# Get unique labels (relations)
labels = sorted(list(set(y_true) | set(y_pred)))

precision, recall, f1, _ = precision_recall_fscore_support(
    y_true, y_pred, average="macro", labels=labels
)
print(f"Macro Precision: {precision:.4f}")
print(f"Macro Recall: {recall:.4f}")
print(f"Macro F1: {f1:.4f}")

# Create figure with two subplots side by side
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 8))

# 1. Regular confusion matrix
cm = confusion_matrix(y_true, y_pred, labels=labels)
```

```
sns.heatmap(
    cm,
    annot=True,
    fmt="d",
    cmap="Blues",
    xticklabels=labels,
    yticklabels=labels,
    ax=ax1,
)
ax1.set_xlabel("Predicted")
ax1.set_ylabel("True")
ax1.set_title("Confusion Matrix for similar classes (Raw Counts)")

# 2. Row-normalized confusion matrix (normalize by true label)
cm_normalized = confusion_matrix(y_true, y_pred, labels=labels,
normalize="true")
sns.heatmap(
    cm_normalized,
    annot=True,
    fmt=".2f",
    cmap="Blues",
    xticklabels=labels,
    yticklabels=labels,
    ax=ax2,
)
ax2.set_xlabel("Predicted")
ax2.set_ylabel("True")
ax2.set_title("Confusion Matrix (Row Normalized)")

plt.tight_layout()
plt.savefig("confusion_matrices.png", dpi=300)
plt.show()
```

```
Accuracy: 0.4608
Macro Precision: 0.3439
Macro Recall: 0.3678
Macro F1: 0.2978
```

```
/home/askatasuna/Документы/HSE/contexts/ContextBench_2025/.venv/lib/
python3.12/site-packages/sklearn/metrics/_classification.py:1565:
UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in
labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```






```
cm_df = pd.DataFrame(cm_normalized, index=labels, columns=labels)

misclassifications = []

for true_class in labels:
    for pred_class in labels:
        if true_class != pred_class:
            raw_count = cm[labels.index(true_class),
labels.index(pred_class)]

            # Only include pairs that actually appeared in the data
            if raw_count > 0:
                misclassifications.append(
                    {
                        "True Class": true_class,
                        "Predicted Class": pred_class,
                        "Error Rate": cm_df.loc[true_class,
pred_class],
                        "Count": raw_count,
                    }
                )

misclass_df = pd.DataFrame(misclassifications)
misclass_df = misclass_df.sort_values(by="Error Rate",
ascending=False)

print("Top confused pairs (by error rate):")
print(misclass_df.head(10))

misclass_df_by_count = misclass_df.sort_values(by="Count",
ascending=False)
print("\nTop confused pairs (by raw count):")
print(misclass_df_by_count.head(10))
```

```
Top confused pairs (by error rate):
                   True Class Predicted Class  Error Rate  Count
30                     effect    cause-effect    1.000000      3
50                  motivation     restatement    1.000000      1
13                       cause    cause-effect    0.797980     79
41                    evidence           cause    0.631579     24
52                 preparation     elaboration    0.630769     41
15               cause-effect           cause    0.545455      6
43  interpretation-evaluation      conclusion    0.500000      6
18                  comparison        contrast    0.482759     14
0                   antithesis      conclusion    0.400000      2
1                   antithesis        contrast    0.400000      2

Top confused pairs (by raw count):
     True Class Predicted Class  Error Rate  Count
13        cause    cause-effect    0.797980     79
48        joint        contrast    0.219880     73
33  elaboration     restatement    0.214511     68
47        joint       condition    0.153614     51
31  elaboration      concession    0.148265     47
52  preparation     elaboration    0.630769     41
34  elaboration    solutionhood    0.094637     30
49        joint        sequence    0.090361     30
41     evidence           cause    0.631579     24
37   evaluation          effect    0.389831     23
```

На этой confusion matrix мы видим большое количество отклонений от диагонали, а также самые частые "смешения" классов. Наиболее часто модель путает классы **cause** - **cause-effect**, **joint** - **contrast**. Также стоит отметить наличие совпадающих классов таких как **effect**, **cause** и **cause-effect**.

Различные классы

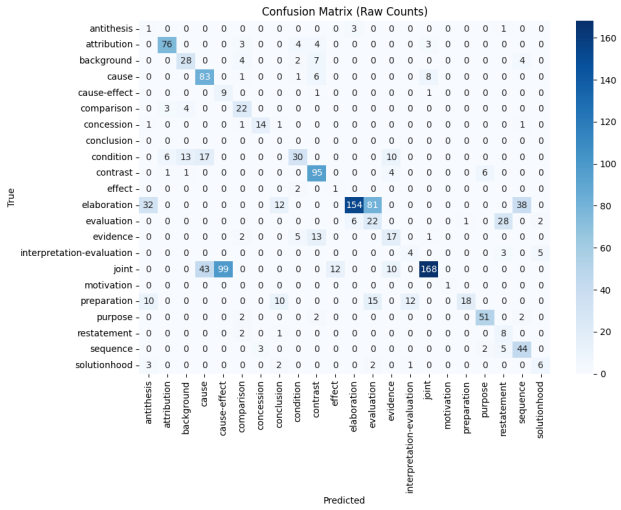
```python
y_true = different_labels_df["correct_answer"].values
y_pred = different_labels_df["answer"].values

# Calculate accuracy
accuracy = accuracy_score(y_true, y_pred)
print(f"Accuracy: {accuracy:.4f}")

# Get unique labels (relations)
labels = sorted(list(set(y_true) | set(y_pred)))

precision, recall, f1, _ = precision_recall_fscore_support(
    y_true, y_pred, average="macro", labels=labels
)
print(f"Macro Precision: {precision:.4f}")
print(f"Macro Recall: {recall:.4f}")
print(f"Macro F1: {f1:.4f}")
```

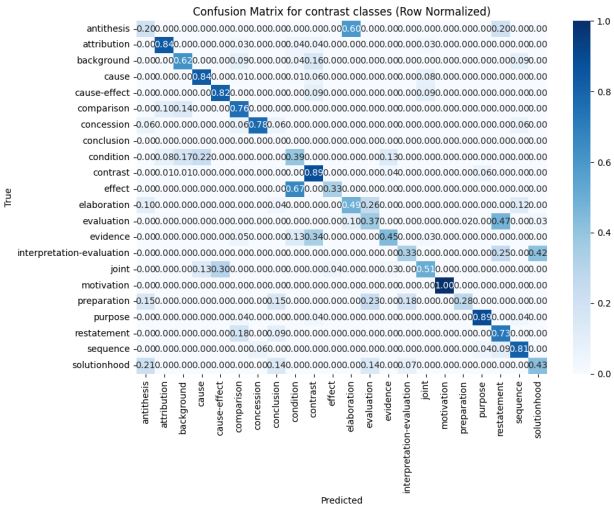
```python
# Create figure with two subplots side by side
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 8))

# 1. Regular confusion matrix
cm = confusion_matrix(y_true, y_pred, labels=labels)
sns.heatmap(
    cm,
    annot=True,
    fmt="d",
    cmap="Blues",
    xticklabels=labels,
    yticklabels=labels,
    ax=ax1,
)
ax1.set_xlabel("Predicted")
ax1.set_ylabel("True")
ax1.set_title("Confusion Matrix (Raw Counts)")

# 2. Row-normalized confusion matrix (normalize by true label)
cm_normalized = confusion_matrix(y_true, y_pred, labels=labels,
normalize="true")
sns.heatmap(
    cm_normalized,
    annot=True,
    fmt=".2f",
    cmap="Blues",
    xticklabels=labels,
    yticklabels=labels,
    ax=ax2,
)
ax2.set_xlabel("Predicted")
ax2.set_ylabel("True")
ax2.set_title("Confusion Matrix for contrast classes (Row
Normalized)")

plt.tight_layout()
plt.savefig("confusion_matrices.png", dpi=300)
plt.show()
```

```
Accuracy: 0.5904
Macro Precision: 0.5340
Macro Recall: 0.5801
Macro F1: 0.5071
```

```
/home/askatasuna/Документы/HSE/contexts/ContextBench_2025/.venv/lib/
python3.12/site-packages/sklearn/metrics/_classification.py:1565:
UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in
labels with no true samples. Use `zero_division` parameter to control
this behavior.
```

```
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```






```
cm_df = pd.DataFrame(cm_normalized, index=labels, columns=labels)

misclassifications = []

for true_class in labels:
    for pred_class in labels:
        if true_class != pred_class:
            raw_count = cm[labels.index(true_class),
labels.index(pred_class)]

            # Only include pairs that actually appeared in the data
            if raw_count > 0:
                misclassifications.append(
                    {
                        "True Class": true_class,
                        "Predicted Class": pred_class,
                        "Error Rate": cm_df.loc[true_class,
pred_class],
                        "Count": raw_count,
                    }
                )

misclass_df = pd.DataFrame(misclassifications)
misclass_df = misclass_df.sort_values(by="Error Rate",
ascending=False)

print("Top confused pairs (by error rate):")
print(misclass_df.head(10))

misclass_df_by_count = misclass_df.sort_values(by="Count",
ascending=False)
```

```
print("\nTop confused pairs (by raw count):")
print(misclass_df_by_count.head(10))
```

```
Top confused pairs (by error rate):
                   True Class Predicted Class  Error Rate  Count
30                     effect       condition    0.666667      2
0                   antithesis     elaboration    0.600000      3
37                  evaluation     restatement    0.474576     28
44  interpretation-evaluation    solutionhood    0.416667      5
41                    evidence        contrast    0.342105     13
46                       joint    cause-effect    0.298193     99
33                 elaboration      evaluation    0.255521     81
43  interpretation-evaluation     restatement    0.250000      3
51                 preparation      evaluation    0.230769     15
24                   condition           cause    0.223684     17

Top confused pairs (by raw count):
     True Class Predicted Class  Error Rate  Count
46         joint    cause-effect    0.298193     99
33   elaboration      evaluation    0.255521     81
45         joint           cause    0.129518     43
34   elaboration        sequence    0.119874     38
31   elaboration      antithesis    0.100946     32
37    evaluation     restatement    0.474576     28
24     condition           cause    0.223684     17
51   preparation      evaluation    0.230769     15
23     condition      background    0.171053     13
41      evidence        contrast    0.342105     13
```

Все классы

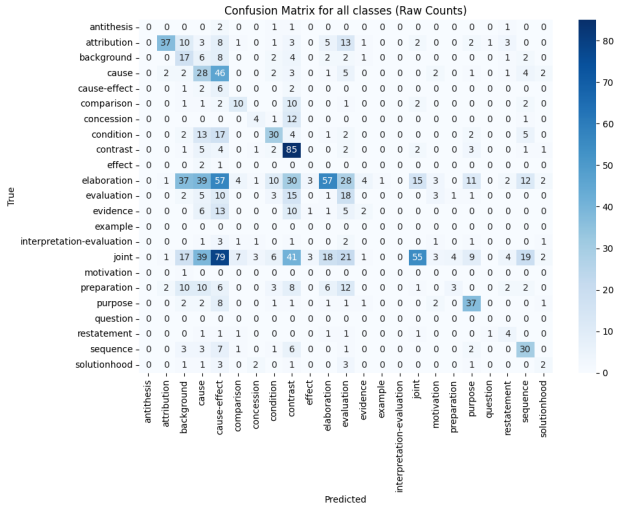
```
y_true = all_labels_df["correct_answer"].values
y_pred = all_labels_df["answer"].values

# Calculate accuracy
accuracy = accuracy_score(y_true, y_pred)
print(f"Accuracy: {accuracy:.4f}")

# Get unique labels (relations)
labels = sorted(list(set(y_true) | set(y_pred)))

precision, recall, f1, _ = precision_recall_fscore_support(
    y_true, y_pred, average="macro", labels=labels
)
print(f"Macro Precision: {precision:.4f}")
print(f"Macro Recall: {recall:.4f}")
print(f"Macro F1: {f1:.4f}")

# Create figure with two subplots side by side
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 8))
```

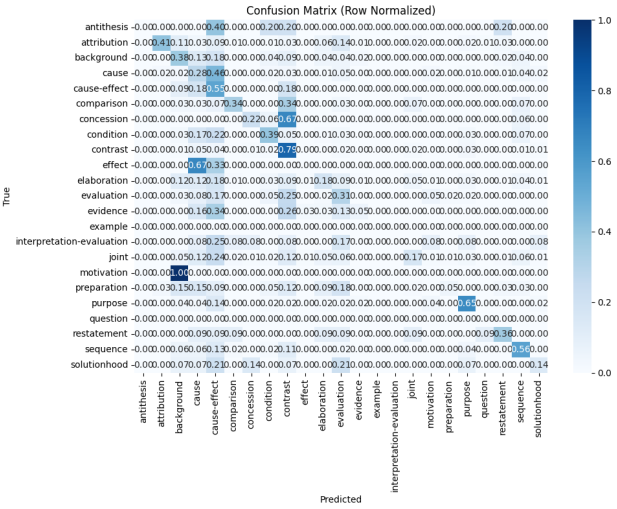
```python
# 1. Regular confusion matrix
cm = confusion_matrix(y_true, y_pred, labels=labels)
sns.heatmap(
    cm,
    annot=True,
    fmt="d",
    cmap="Blues",
    xticklabels=labels,
    yticklabels=labels,
    ax=ax1,
)
ax1.set_xlabel("Predicted")
ax1.set_ylabel("True")
ax1.set_title("Confusion Matrix for all classes (Raw Counts)")

# 2. Row-normalized confusion matrix (normalize by true label)
cm_normalized = confusion_matrix(y_true, y_pred, labels=labels,
normalize="true")
sns.heatmap(
    cm_normalized,
    annot=True,
    fmt=".2f",
    cmap="Blues",
    xticklabels=labels,
    yticklabels=labels,
    ax=ax2,
)
ax2.set_xlabel("Predicted")
ax2.set_ylabel("True")
ax2.set_title("Confusion Matrix (Row Normalized)")

plt.tight_layout()
plt.savefig("confusion_matrices.png", dpi=300)
plt.show()
```

```
Accuracy: 0.2945
Macro Precision: 0.2667
Macro Recall: 0.2536
Macro F1: 0.2167
```

```
/home/askatasuna/Документы/HSE/contexts/ContextBench_2025/.venv/lib/
python3.12/site-packages/sklearn/metrics/_classification.py:1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
/home/askatasuna/Документы/HSE/contexts/ContextBench_2025/.venv/lib/
python3.12/site-packages/sklearn/metrics/_classification.py:1565:
```

```
UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in
labels with no true samples. Use `zero_division` parameter to control
this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```





Поиск корреляций

task	precision
contrasting	0.4975
similar	0.4135
random	0.3580
all	0.2261

Итак, мы увидели, что precision в целом согласуется с нашими ожиданиями: наибольшее значение precision у сабсета с наиболее различными классами, что говорит о том, что модель с бОльшим успехом различает классы менее похожи по косинусной близости. Проверим эти данных также с помощью корреляции precision и среднего косинусного расстояния между лейблом и вариантами ответа в сабсете.

```
similarities_df.head()
```

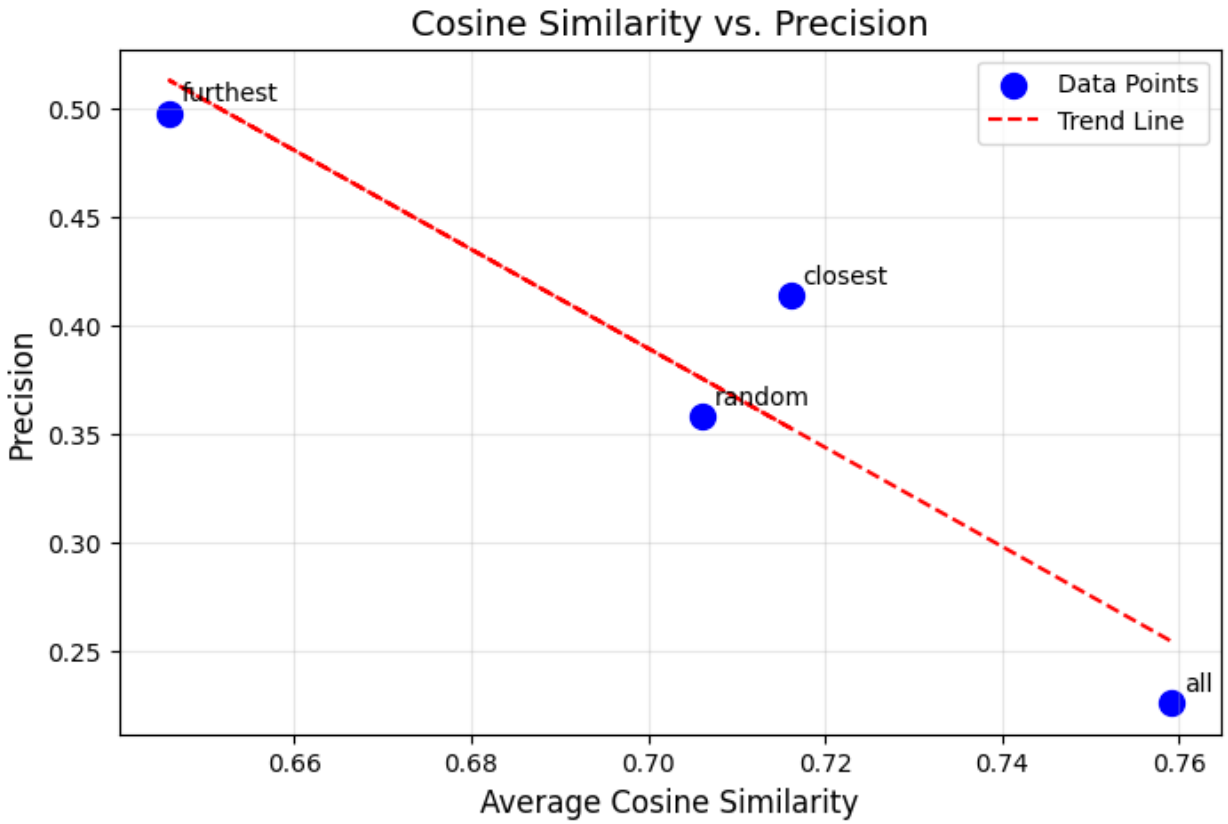
```
          Average Cosine Similarity
closest                    0.716062
furthest                   0.645864
random                     0.706171
all                        0.759086
```

```
similarities_df['Precision'] = [0.4135, 0.4975, 0.3580, 0.2261]
similarities_df.head()
```

```
          Average Cosine Similarity  Precision
closest                    0.716062     0.4135
```

```
furthest                    0.645864     0.4975
random                      0.706171     0.3580
all                         0.759086     0.2261
```

```
similarities_df['Group'] = ["closest", "furthest", "random", "all"]

plt.figure(figsize=(8, 5))
plt.scatter(
    similarities_df['Average Cosine Similarity'],
    similarities_df['Precision'],
    color='blue',
    s=100,
    label='Data Points'
)

plt.xlabel('Average Cosine Similarity', fontsize=12)
plt.ylabel('Precision', fontsize=12)
plt.title('Cosine Similarity vs. Precision', fontsize=14)

for i, row in similarities_df.iterrows():
    plt.annotate(row['Group'], (row['Average Cosine Similarity'],
row['Precision']),
                 textcoords="offset points", xytext=(5,5), ha='left')

import numpy as np
x = similarities_df['Average Cosine Similarity']
y = similarities_df['Precision']
m, b = np.polyfit(x, y, 1)
plt.plot(x, m*x + b, color='red', linestyle='--', label='Trend Line')

plt.legend()
plt.grid(True, alpha=0.3)
plt.show()
```




```
corr, p_value = pearsonr(similarities_df['Average Cosine Similarity'],
similarities_df['Precision'])
print(f"Pearson correlation: {corr:.3f}, p-value: {p_value:.4f}")
```

```
Pearson correlation: -0.933, p-value: 0.0669
```

Итак, корреляция есть, и она согласуется с гипотезой H0, но с высоковатым p-value, что говорит о том, что мы можем отвергнуть нулевую гипотезу только увеличив a. Однако, мы можем предположить, что модель всё же получает дополнительную информацию о правильном ответе из других вариантов ответа, что отражается на precision.

Выводы

В ходе эксперимента мы проверили гипотезу о том, что семантическая близость классов обратно пропорциональна precision LLM классификатора. Мы собрали 4 варианта датасета, различающихся только набором классов подаваемых модели на вход для выбора, и проскорили модель gpt-4o-mini на каждом из этих датасетов. Результаты показали, что precision в целом согласуется с нашими ожиданиями: наибольшее значение precision у сабсета с наиболее различными классами (эмбеддинги от BGE-small), что говорит в пользу гипотезы H0. Однако, p-value достаточно высокое, что требует дальнейшего исследования с большим количеством вариантов датасетов и моделей.