Алайнмент T-lite под агентские кейсы

AnomalyDetection

МИРОНОВ ВЛАДИМИР (DS, ANALYTICS)

АРТЁМ АКОПЯН (DS, ANALYTICS)

ГОРДЕЕВ ЕВГЕНИЙ (DS, ANALYTICS)

ШУМСКИЙ ИГОРЬ (DS, ANALYTICS)

СОЛОНЧЕНКО РОМАН (VIUSALIZATION, DATABASE)

Цели и задачи на хакатон

- Проанализировать задачу, и понять что от нас требуется со стороны бизнеса
- Построить бенчмарк, который бы всесторонне оценивал датасет исходя из его структуры
- Настрооить модель T-Lite и Aligment исходя из поставленной задачи
- Настроить телеграмм -бот, REST-API согласно контракту
- Постараться учесть в генерируемых датасетах вариативную природу данных
- Разработать решение максимально быстрое по скорости обработки и качеству ответов

Генерация датасета

Генерация датасета включала в себя использование решений по разлиыным моделям: T5-large, BART-large, XLNet, SDGX, ... Пример исходного датасета показан слева и сгенерированного справа.

Было проведено множество тестов, для оценки генерируемых даатсетов, наилучший вариант показала SDGX.

Далее полученный датасет затем проверялся на многочисленных тестах на нашем бенчмарке.

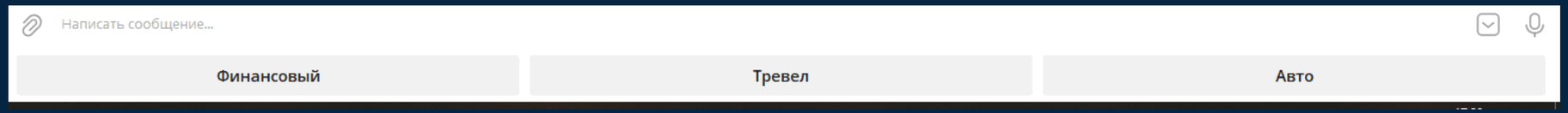
	role	content
0	bot	Текущие котировки акций Apple составляют $173.21.
1	user	Какая доходность облигаций государственного за...
2	bot	На изменение валютного курса влияют экономичес...
3	user	Как выбрать хорошего финансового консультанта?
4	user	Какой прогноз по цене нефти на следующий квартал?
5	bot	Сегодня на фондовом рынке наблюдается рост инд...
6	user	Какие риски связаны с инвестированием в стартапы?
7	user	Каков прогноз по цене золота на следующий месяц?
8	user	Какой текущий курс биткойна к доллару США?
9	bot	В этом квартале дивиденды выплатят компании Mi...

```
     role                                                content
0     bot  Риски инвестирования в стартапы включают высок...
1     bot   Каков прогноз по цене золота на следующий месяц?
2    user   Каков прогноз по цене золота на следующий месяц?
3    user  Новости из Китая, такие как изменения в эконом...
4    user  Какие акции торгуются на Нью-Йоркской фондовой...
..    ...                                                ...
995  user  Текущий тренд на рынке недвижимости показывает...
996  user  Курс евро к доллару изменился с 1.092 до 1.095...
997  user  На этой неделе ожидаются отчеты по безработице...
998   bot  Ставки по ипотечным кредитам в этом месяце сни...
999   bot  Криптовалюта — это цифровая валюта, использующ...

[1000 rows x 2 columns]
```

Ассистент (Артем)

Rest API



Ассистент (Евгений)

Телеграмм бот

Оценка датасета на бенчмарке

```python
def dataset_complexity(data):
    text_lengths = [len(tokenizer.tokenize(text)) for text in data['content']]
    return {
        'avg_length': np.mean(text_lengths),
        'max_length': np.max(text_lengths),
        'min_length': np.min(text_lengths),
        'length_variability': np.std(text_lengths)
    }

complexity_stats = dataset_complexity(sampled_data)
print(f"Сложность датасета: {complexity_stats}")
```

```
Сложность датасета: {'avg_length': 21.746, 'max_length': 41, 'min_length': 12, 'length_variability': 9.116769383942977}
```

1. Оценка сложности данных

```python
def token_stats(data):
    all_tokens = [token for text in data['content'] for token in tokenizer.tokenize(text)]
    token_counter = Counter(all_tokens)
    return {
        'num_unique_tokens': len(token_counter),
        'top_10_tokens': token_counter.most_common(10)
    }

token_stats_train = token_stats(sampled_data)
print(f"Токенизация данных: {token_stats_train}")
```

```
Токенизация данных: {'num_unique_tokens': 331, 'top_10_tokens': [('▁', 2237), ('.', 625), ('и', 542), ('е', 530),
```

2. Оценка разнообразия токенов

```python
model = SentenceTransformer('paraphrase-MiniLM-L6-v2')

def semantic_diversity(data, sample_size=100):
    sample_texts = data['content'][:sample_size]
    embeddings = model.encode(sample_texts)
    similarity_matrix = cosine_similarity(embeddings)

    avg_similarity = similarity_matrix.mean()
    max_similarity = similarity_matrix.max()

    return avg_similarity, max_similarity

avg_sim, max_sim = semantic_diversity(sampled_data)
print(f"Среднее семантическое сходство: {avg_sim}, Максимальное семантическое сходство: {max_sim}")
```

```
modules.json: 100% 229/229 [00:00<00:00, 6.41kB/s]
config_sentence_transformers.json: 100% 122/122 [00:00<00:00, 5.16kB/s]
README.md: 100% 3.73k/3.73k [00:00<00:00, 95.5kB/s]
sentence_bert_config.json: 100% 53.0/53.0 [00:00<00:00, 2.11kB/s]
config.json: 100% 629/629 [00:00<00:00, 41.3kB/s]
model.safetensors: 100% 90.9M/90.9M [00:01<00:00, 112MB/s]
```

3. Семантическая оценка данных

```python
def find_semantic_duplicates(data, threshold=0.9):
    embeddings = model.encode(data['content'])
    similarity_matrix = cosine_similarity(embeddings)

    duplicate_pairs = []
    for i in range(len(similarity_matrix)):
        for j in range(i+1, len(similarity_matrix)):
            if similarity_matrix[i, j] > threshold:
                duplicate_pairs.append((i, j))

    return duplicate_pairs

sem_duplicates = find_semantic_duplicates(sampled_data, threshold=0.9)
print(f"Найдено семантических дубликатов: {len(sem_duplicates)}")
```

```
Найдено семантических дубликатов: 102736
```

4. Дублирование на уровне смысла

Оценка датасета на бенчмарке

```python
def rare_tokens_analysis(data, threshold=5):
    token_counts = Counter([token for text in data['content'] for token in tokenizer.tokenize(text)])
    rare_tokens = {token: count for token, count in token_counts.items() if count < threshold}
    return rare_tokens

rare_tokens = rare_tokens_analysis(sampled_data)
print(f"Редкие токены (встречаются меньше раз): {len(rare_tokens)}")

nlp = spacy.load("en_core_web_sm")

def named_entity_analysis(data):
    entity_counts = Counter()
    for text in data['content']:
        doc = nlp(text)
        for ent in doc.ents:
            entity_counts[ent.text] += 1
    return entity_counts

entity_stats = named_entity_analysis(sampled_data)
print(f"Самые частые сущности: {entity_stats.most_common(10)}")
```

```
Редкие токены (встречаются меньше раз): 0
Самые частые сущности: [('Какие', 229), ('США', 97), ('Microsoft', 70), ('экономические', 64), ('по', 61), ('кредитам
```

5. Анализ редкости

```python
def coherence_score(data, sample_size=100):
    sample_texts = data['content'][:sample_size]
    coherence_scores = []

    for text in sample_texts:
        sentences = text.split('. ')
        if len(sentences) > 1:
            sentence_embeddings = model.encode(sentences)
            sentence_similarities = cosine_similarity(sentence_embeddings)
            coherence_scores.append(sentence_similarities.mean())

    return np.mean(coherence_scores)

avg_coherence = coherence_score(sampled_data)
print(f"Средняя когерентность текстов: {avg_coherence}")
```

```
Средняя когерентность текстов: 0.9242082834243774
```

6. Оценка когерентности данных

```python
toxicity_classifier = pipeline('text-classification', model='unitary/toxic-bert')
toxicity_results = sampled_data['content'].apply(lambda text: toxicity_classifier(text)[0])
print(f"Результаты токсичности:\n{toxicity_results}")
```

```
config.json: 100% 811/811 [00:00<00:00, 29.0kB/s]
model.safetensors: 100% 438M/438M [00:09<00:00, 87.8MB/s]
tokenizer_config.json: 100% 174/174 [00:00<00:00, 3.89kB/s]
vocab.txt: 100% 232k/232k [00:00<00:00, 3.52MB/s]
special_tokens_map.json: 100% 112/112 [00:00<00:00, 4.75kB/s]
```

7. Оценка токсичности текста

```python
model = SentenceTransformer('paraphrase-MiniLM-L6-v2')

def detect_ambiguous_texts(data, n_clusters=10):
    embeddings = model.encode(sampled_data['content'])
    kmeans = KMeans(n_clusters=n_clusters)
    clusters = kmeans.fit_predict(embeddings)

    ambiguous_texts = []
    for i in range(n_clusters):
        cluster_texts = [data['content'][j] for j in range(len(data['content'])) if clusters[j] == i]
        if len(cluster_texts) > 1:
            ambiguous_texts.append(cluster_texts)

    return ambiguous_texts

ambiguous_clusters = detect_ambiguous_texts(sampled_data)
print(f"Найдено {len(ambiguous_clusters)} неоднозначных кластеров.")
```

```
Найдено 10 неоднозначных кластеров.
```

8. Оценка запутанности текста

Оценка датасета на бенчмарке

```
Распределение по классам 'role':
role
bot     612
user    388
Name: count, dtype: int64

Процентное соотношение классов:
role
bot     61.2
user    38.8
Name: proportion, dtype: float64

Частота ключевых слов в контенте:
{'акции': 134, 'дивиденды': 57, 'валюта': 57, 'биткойн': 43, 'кредит': 61}

Среднее количество уникальных слов в текстах:
10.431
```

9. Проверка баланса данных

```
Типы предложений (Утвердительные/Вопросительные):
sentence_type
Утвердительное    576
Вопросительное    424
Name: count, dtype: int64

Временные формы (Настоящее/Прошедшее/Будущее):
tense
Неопределённое время    869
Настоящее время         131
Name: count, dtype: int64

Сложность предложений (Простые/Сложные):
sentence_complexity
Простое предложение    587
Сложное предложение    413
Name: count, dtype: int64
```

10. Оценка языковой вариативности

```
Наличие HTML-тегов:
contains_html
False    1000
Name: count, dtype: int64

Наличие нестандартных символов:
contains_nonstandard_symbols
False    695
True     305
Name: count, dtype: int64

Сломанные предложения (нет завершающего знака препинания):
is_broken_sentence
False    1000
Name: count, dtype: int64

Наличие дублирующихся пробелов:
contains_extra_spaces
False    1000
Name: count, dtype: int64
```

11. Оценка структуры и формата текста

```python
tool = language_tool_python.LanguageTool('ru')

def grammar_check(text):
    matches = tool.check(text)
    return len(matches)  # Возвращаем количество найденных ошибок

sampled_data['grammar_errors'] = sampled_data['content'].apply(grammar_check)

print("Количество грамматических ошибок в каждом тексте:")
print(sampled_data[['content', 'grammar_errors']])
average_errors = sampled_data['grammar_errors'].mean()
print(f"\nСреднее количество ошибок на текст: {average_errors}")
```

```
Downloading LanguageTool 6.4: 100%|██████████| 246M/246M [00:06<00:00, 37.3MB/s]
INFO:language_tool_python.download_lt:Unzipping /tmp/tmp69vy_jtz.zip to /root/.cache/language_tool_python.
INFO:language_tool_python.download_lt:Downloaded https://www.languagetool.org/download/LanguageTool-6.4.zip
```

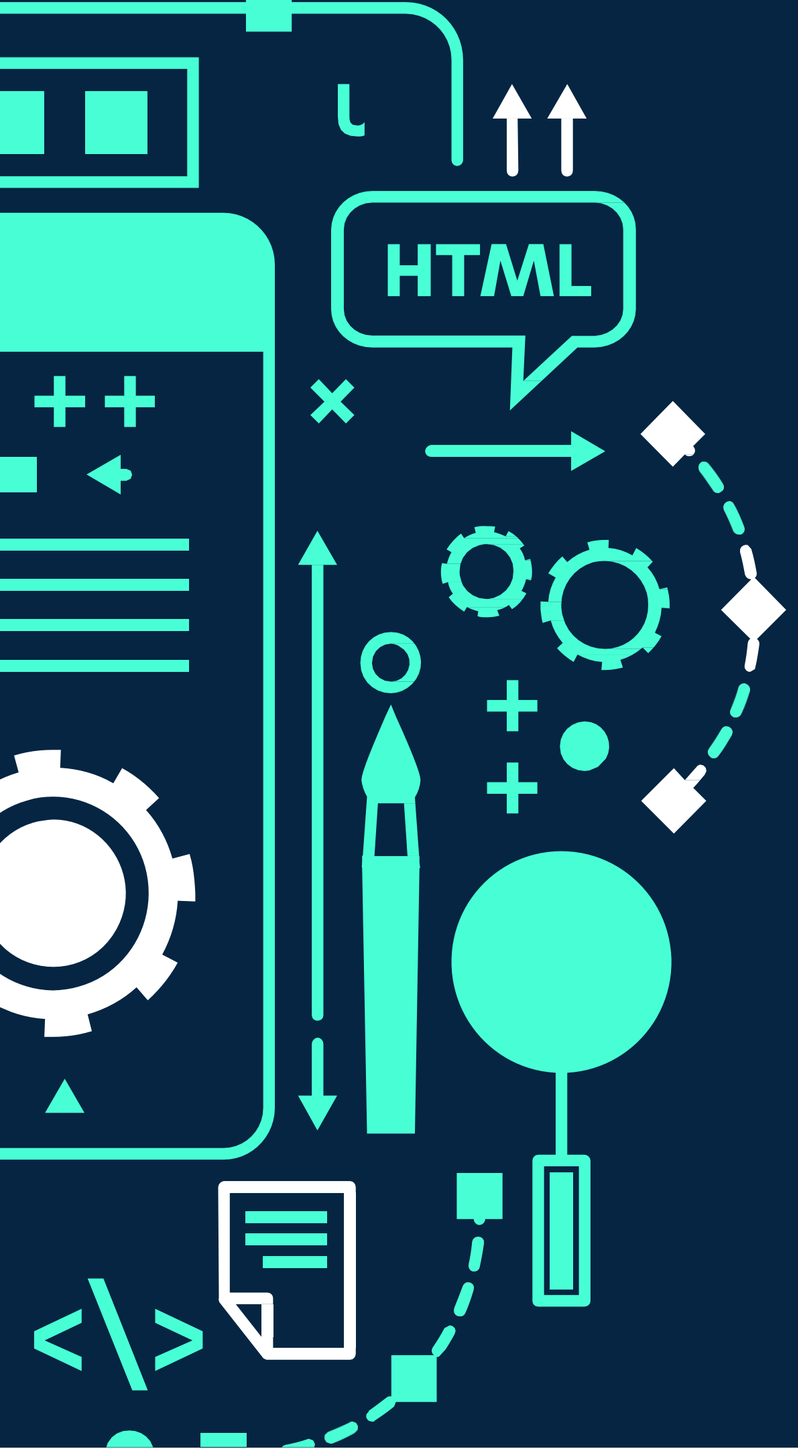
12. Оценка грамматической правильности

Оценка датасета на бенчмарке

13. Оценка эмоциональной окраски
14. Оценка референтности (Fact-Cheking)
15. Оценка стереотипов и предвзятости
16. Анализ контекстной зависимости
17. Оценка задачи на многоязычность (Multilingual Task Evaluation)
18. Анализ контекстной зависимости
19. Оценка задачи на многоязычность (Multilingual Task Evaluation)
20. Проверка на нелогичные данные и проверка баланса данных

Планы

- Улучшить LLM Benchmarker Модель
- Улучшить интерфейс telegram-бота
- Собрать больше датасетов
- Увеличить кол-во ассистентов



Спасибо за внимание!