Changellenge >> Cup IT 2023

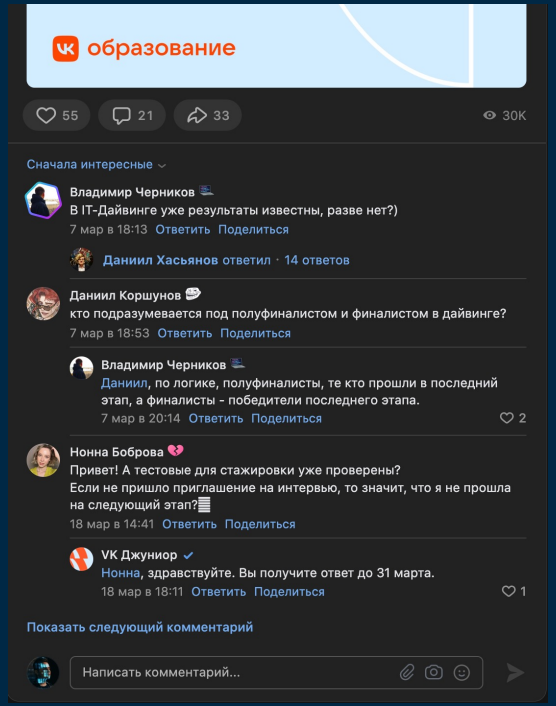
Data Science

Команда: MLOops

Анализ кейса

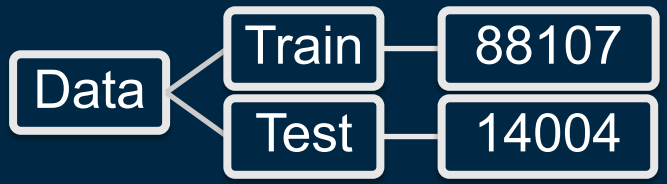
- Персонифицированность социальных сетей – мировой тренд, помогающий пользователям получать релевантную, интересную для них информацию
- Новостная лента – привычное «место» для любого человека 21 века
- Современные методы машинного обучения и анализа данных позволяют получить релевантный для пользователя контент

Анализ кейса

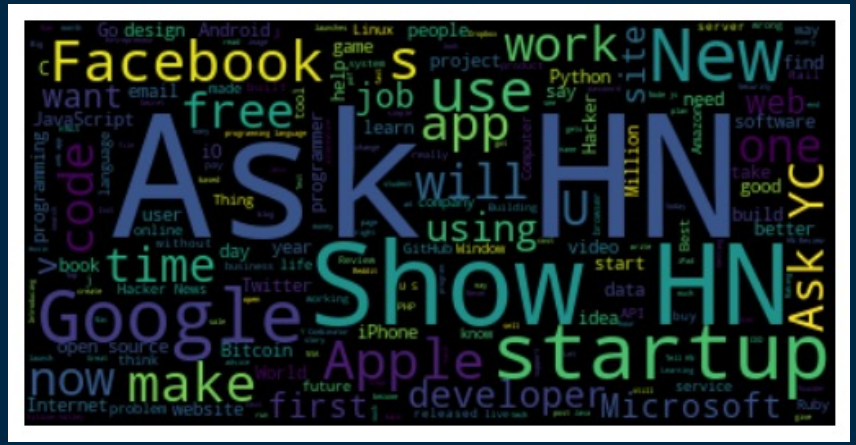


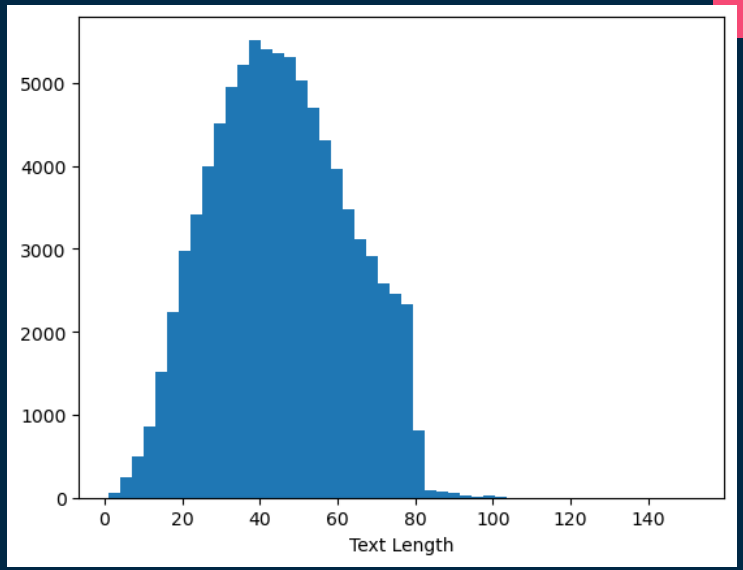
- Комментарии – способ пользователя оценить контент поста, задать вопрос или дать полезную информацию
- Часто комментарии не имеют релевантной «нагрузки», поэтому есть необходимость показывать их в определенном упорядоченном порядке
- Есть несколько решений, удовлетворяющих потребности пользователей в релевантном контенте

Решение (EDA)



Text length stats:
mean = 46, std = 17
Comment length stats:
mean = 523, std = 595





Решение (Выбор метода)

Для решения кейса мы решили использовать **TF-IDF**

TF-IDF (от англ. TF – term frequency, IDF — inverse document frequency) – статистическая мера, используемая для оценки важности слова в контексте документа, являющегося частью коллекции документов или корпуса.

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

Решение (Тренировка и валидация)

- Мы использовали простейшую модель линейной регрессии и TfidfVectorizer
- В качестве метрики оценки мы использовали Normalized Discounted Cumulative Gain (NDCG)

```python
# Extract features train data
X_train = []
y_train = []
for sample in train_data:
    post = sample['text']
    comments = sample['comments']
    comment_texts = [comment['text'] for comment in comments]
    comment_scores = [comment['score'] for comment in comments]
    X_train.append(post + ' '.join(comment_texts))
    y_train.append(comment_scores)

# Extract features from test
X_test = []
for sample in test_data:
    post = sample['text']
    comments = sample['comments']
    comment_texts = [comment['text'] for comment in comments]
    X_test.append(post + ' '.join(comment_texts))
```

```python
# Vectorize text
vectorizer = TfidfVectorizer() # vectorizes them using TF-IDF
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

# Train model
model = LinearRegression()
model.fit(X_train, y_train)

# predict
y_test_pred = model.predict(X_test)

# Fill null score values in test with predicted values
for i in range(len(test_data)):
    for j in range(5):
        if test_data[i]['comments'][j]['score'] is None:
            test_data[i]['comments'][j]['score'] = int(y_test_pred[i][::-1][j])

# Compute NDCG  on test
y_test_true = [[comment['score'] for comment in sample['comments']] for sample in test_data]
ndcg = ndcg_score(y_test_true, y_test_pred, k=5)
```

NDCG score на тестовых данных: 0.6104173580303227

Анализ и выводы

- Получившийся скор показывает то, что выбранная стратегия справляется с поставленной задачей ранжирования комментариев по популярности
- При более тщательном feature-engineering, подборе более сложной модели и ее тюнинге могут получиться результаты и скор, выше представленного
- Как один из перспективных вариантов можно рассмотреть методы, применяемые при разработке рекомендательных систем – использования моделей UserKNN (User-Item), различных матричных разложений, что позволит получать релевантные ранжирования комментариев для каждого отдельного пользователя