Федеральное государственное автономное образовательное
учреждение высшего образования

Национальный исследовательский университет
«Высшая школа экономики»

Факультет компьютерных наук

Основная образовательная программа
Прикладная математика и информатика

ПРОЕКТ ПО КУРСУ АВТОМАТИЧЕСКАЯ ОБРАБОТКА ТЕКСТОВ

«Clickbait Challenge at SemEval 2023 - Clickbait Spoiling»

Выполнили студенты:
Соколов Ян, группа 194, 4 курс,
Екимов Егор, группа 194, 4 курс
Токкожин Арсен, группа 194, 4 курс
Гвасалия Лукас, группа 194, 4 курс

МОСКВА 2022

СОДЕРЖАНИЕ

1 Краткое описание задачи

1.1 Задача

Посты-кликбейты ссылаются на веб-страницы и рекламируют их содеражание, вызывая у пользователя любопытство, а не предоставляя информативные резюме. Основная задача заключается в классификации типа спойлера, который предупреждает пост-кликбейт.

1.2 Данные

Датасет содержит посты кликбейты, а также предобрабоанные документы, на которыее ссылается данный кликбейт. Также спойлеры делятся на три типа: спойлеры коротких фраз, спойлеры более длинных отрывков и спойлеры нескольких непоследовательных фрагментов текст („phrase“„passage“„multi“).

Всего у нас имеется 3200 постов для обучения и 800 для валидации, а также 1000 постов на которых в конечном итоге будет проверен наш классификатор. Данные представляют собой следующий JSON формат:

- uuid: uuid поста.
- postText: текст поста кликбейта. targetParagraphs: основное содержание связанной с постом страницы, основной текст выделен вручную.
- targetTitle: название связанной с постом страницы.
- targetUrl: url страницы на которую ссылается кликбейт-пост.
- humanSpoiler: сгенерированный человеком спойлер (абстракция) для поста-кликбейта со страницы, на которую дана ссылка.
- spoiler: извлеченный человеком спойлер для кликабельного поста со связанной веб-страницы.
- spoilerPositions: позиция спойлера в тексте, извлеченного человеком для кликабельного поста со связанной веб-страницы.
- tags: тип спойлера (может быть "фраза "отрывок"или "мульти"), который должен быть классифицирован.
- некоторые поля содержат дополнительную метаинформацию о записи, но не используются: postId, postPlatform, targetDescription, targetKeywords, targetMedia.

Ниже приведен пример одного из постов в нашем датасете.


```
{
  "uuid": "0af11f6b-c889-4520-9372-66ba25cb7657",
  "postText": ["Wes Welker Wanted Dinner With Tom Brady, But Patriots QB Had Better Idea"],
  "targetParagraphs": [
    "It’ll be just like old times this weekend for Tom Brady and Wes Welker.",
    "Welker revealed Friday morning on a Miami radio station that he contacted Brady because he’ll be in town for Sunday’s game between the New England Patriots and Miami Dolphins at Gillette Stadium. It seemed like a perfect opportunity for the two to catch up.",
    "But Brady’s definition of \"catching up\" involves far more than just a meal. In fact, it involves some literal \"catching\" as the Patriots quarterback looks to stay sharp during his four-game Deflategate suspension.",
    "\"I hit him up to do dinner Saturday night. He’s like, ‘I’m going to be flying in from Ann Arbor later (after the Michigan-Colorado football game), but how about that morning we go throw?’ \" Welker said on WQAM, per The Boston Globe. \"And I’m just sitting there, I’m like, ‘I was just thinking about dinner, but yeah, sure. I’ll get over there early and we can throw a little bit.’ \"",
    "Welker was one of Brady’s favorite targets for six seasons from 2007 to 2012. It’s understandable him and Brady want to meet with both being in the same area. But Brady typically is all business during football season. Welker probably should have known what he was getting into when reaching out to his buddy.",
    "\"That’s the only thing we really have planned,\" Welker said of his upcoming workout with Brady. \"It’s just funny. I’m sitting there trying to have dinner. ‘Hey, get your ass up here and let’s go throw.’ I’m like, ‘Aw jeez, man.’ He’s going to have me running like 2-minute drills in his backyard or something.\"",
    "Maybe Brady will put a good word in for Welker down in Foxboro if the former Patriots wide receiver impresses him enough."
  ],
  "targetTitle": "Wes Welker Wanted Dinner With Tom Brady, But Patriots QB Had A Better Idea",
  "targetUrl": "http://nesn.com/2016/09/wes-welker-wanted-dinner-with-tom-brady-but-patriots-qb-had-better-idea/",
  "spoiler": ["how about that morning we go throw?"],
  "spoilerPositions": [[[3, 151], [3, 186]]],
  "tags": ["passage"]
}
```


1.3 Анализ данных

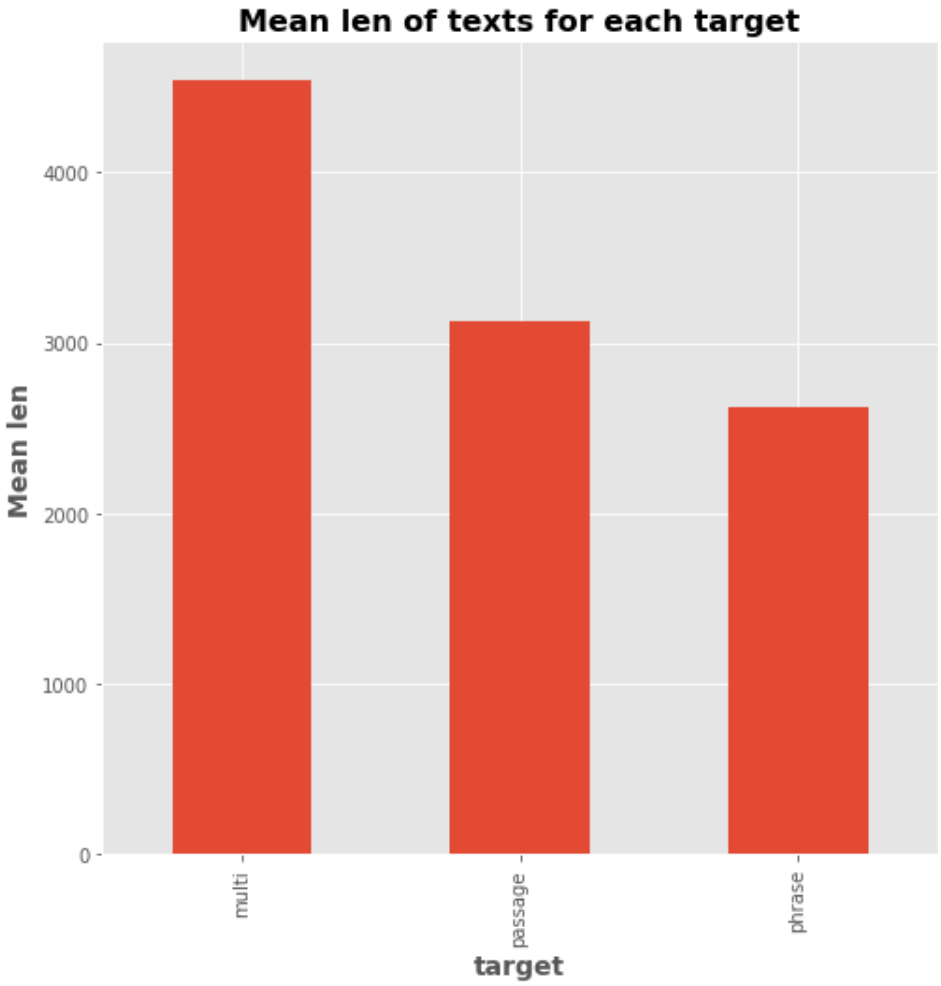
В приложенной к этому отчету тетрадке мы провели анализ предоставленных данных, заметили, что количество классов не сбалансировано:

- phrase -> 1367
- passage -> 1274
- multi -> 559

Рассмотрели топ 5 часто встречающихся слов в каждом из классов:

target		text_tokenized
multi	's	2542
	one	1403
	n't	1205
	people	1177
	like	1071
passage	's	4477
	one	1986
	said	1871
	inc.	1854
	like	1656
phrase	's	4587
	one	1920
	said	1915
	n't	1533
	like	1412

Построили график средней длинны текста в зависимости от класса его спойлера:



Средняя длина текста ментше в фразах и больше в multi, что вполне закономерно. Опечаток в текстах постов, а также примеры некорректной разметки мы не нашли.

2 Ход работы

2.1 Baselines

В соревновании организаторами был предоставлен код для получения двух baseline-ов:

- Baseline1 - предсказние одного класса "passage"
- Baseline2 - предсказания, полученные с помощью fine-tuned трансформера

В качестве метрики для оценивания качества мы решили использовать F-меру.

f-мера для baseline1 = 0.2

f-мера для baseline2 = 0.73

2.2 Базовая архитектура модели

В качестве базовой модели мы решил попробовать fasttext так как эта модель является довольно таки простой и хорошей чтобы использовать ее в качестве baseline модели.Далее мы предварительно предобработали данные: применили токенизацию, учли при токенизации стоп слова, определенные знаки препинания, а также пунктуацию. Также мы осуществили подбор гиперпараметра который отвечат за количество эпох, которое будет обучаться наш классификатор. Итоговый скор на f-мере получился 0,47. Таким образом нам удалось побить первый baseline. В дальнейшем мы планируем использовать более сложные архитектуры: BERT, RoBERTa. Сравнить полученные результаты и выбрать лучшую модель. В изученных нами статьях очень хорошо себя проявила модель RoBERTa.

2.3 Результаты

На данный момент у нас отсуствтует доступ к соревнованию, чтобы сделать submit. Так как нам не удалось сделать пока еще submit, мы добавили k-fold cross-validation, мы взяли k=5. Итоговый скор на f-мере получился 0,47. Таким образом нам удалось побить первый baseline.

2.4 Краткий обзор литературы

Clickbait Spoiling via Question Answering and Passage Retrieval

В данной статье авторы предлагают свои методы решения задачи классификации типа спойлера, который предупреждает пост-кликбейт. Авторы представили следующие этапы методологии, которые они использовоали в своей работы:

- Сбор данных.Для получения датасета, авторы использовали 5 аккаунтов в 3 социальных сетях: Twitter, Redis, Facebook (*запрещенная в России террористическая организация).
- Обработка и предварительный анализ данных.В качестве моделей были рассмотрены классические модели: Naive Bayes, Logistic Regression, SVM. Также были использованы нейроные сети: BERT, DeBERTa, RoBERTa.
- Выбор модели, а также подбор гиперпараметров для улучшения качества.
- Оценка качества моделей.

Авторы провели три экмперимента:

- multi-class
- one-vs-rest
- one-vs-one

Результаты первых двух экспериментов представленны в таблице 2

Table 2: Effectiveness of spoiler type classification in the multi-class (first column) and one-vs-rest settings on 1000 test posts (training: 3200; validation: 800).

Model	**Balanced accuracy** (0, 1, 2 indicate class labels)			
Phrase	0	1	0	0
Passage	1	0	1	0
Multipart	2	0	0	1
Naïve Bayes	56.15	65.03	62.50	64.82
SVM	59.62	68.03	68.70	70.28
Log. Regression	60.04	68.04	69.33	71.26
BERT	67.84	74.06	75.70	75.56
DeBERTa	**73.63**	78.39	78.65	77.93
RoBERTa	71.57	**80.39**	**79.30**	**79.12**

В итоге в multi-class задаче - лучше всех себя показала модель DeBERTa. В задаче one-vs-rest - RoBERTa. Результаты последнего эксперимента представленны в таблице 3

Table 3: Effectiveness of spoiler type classification in the one-vs-one (phrase-vs-passage) setting on 826 test posts (training: 2,641; validation: 657).

Model	Effectiveness				
	TP	TN	FP	FN	Acc.
Naïve Bayes	298	256	147	125	67.07
SVM	311	264	139	112	69.61
Log. Regression	306	273	130	117	70.10
BERT	315	315	88	108	76.27
DeBERTa	318	335	68	105	79.06
RoBERTa	332	332	71	91	**80.39**

Здесь лучше всего себя проявила архитектура модели RoBERTa Также авторы особо подмечают, что во всех экспериментах нейронные модели показывают результаты лучше, чем те, что дают модели основанные на классических подходах.

RoBERTa: A Robustly Optimized BERT Pretraining Approach

В данной статье авторы представляют оптимизированную версию известной модели BERT. В статье были описаны методы, с помощью которых старая модель была улучшена. Поговорим о некоторых из них:

- RoBERTa обучена на 160GB данных, в то время как BERT - 16GB
- Данная модель обучена на 500000 шагах, что превосходит старую модель
- Использовано Dynamic Masking
- Использовано кодирование текста Byte-Pair Encoding

Посмотрим на результаты:

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

Видно, что RoBERTa превосходит BERT по всем параметрам

3 Контрольная точка 2

В предыдущем КТ1 мы остановились на том что побили базовый baseline с помощью fasttext. Вспомним, что качество оценивается на основе f1 меры.

$f1_{fasttext} = 0.46$

$f1_{baseline_1} = 0.19$

$f1_{baseline_2} = 0.73$

Однако для того чтобы преодолеть второй baseline этого оказалось недостаточно. Проанализировав описанные в предыдущей части отчета источники, мы пришли к выводу, что стоит попробовать использовать более сложную архитектуру - transformer. А именно стоит использовать BERT и модели основанные на ее архитектуре, такие как: RoBERTa, DeBERTa, ALBERT и т.д.

3.1 Новая архитектура модели

Если взглянуть на результаты работы различных моделей из статьи Clickbait Spoiling via Question Answering and Passage Retrieval то очень хорошо видно что нейросетевые архитектуры проявляют себя лучше, чем классические подходы. Поэтому мы решили тоже попробовать нейросетевые модели основанные на архитектуре трансформеров. Впервые данная архитектура была представленна в статье Attention is All You Need. В основе этой архитектуры лежит так называемый механизм внимания(attention), заметно улучшающий показатели модели. Исходя из документации библиотеки fasttext, они используют под капотом логичтическую регрессию для задачи классификации текста. Мы же собираемся использовать нейросетевую архитектуру, которая по своей структуре может рассматривать намного большее количество признаков и зависимостей. Также мы используем механизм внимания. Данный механизм значительно улучшает качество систем машинного перевода, позволяя моделям концентрироваться на релевантных частях входных последовательностей.

В статье Clickbait Spoiling via Question Answering and Passage Retrieval авторы рассмотрели только 3 архитектуры трансформеров: BERT, RoBERTa, DeBERTa. Мы же пошли дальше и решили поэксперементировать с другими моделями: ALBERT, DistilBERT, XLNET.

BERT - языковая модель, основанная на архитектуре transformer. На момент создания данная модель демонстрировала высокие результаты и являлась state-of-the-art model. Однако сейчас существует несколько модификаций данной модели: ALBERT, RoBERTa, DeBERTa, DistilBERT. Идея в основе BERT лежит очень простая: на вход нейросети подают фразы, в которых 15 процентов слов заменены на [Mask]. Тогда наша нейросеть должна предсказывать эти закрытые маской слова. Сам по себе BERT представляет собой нейронную сеть, основу которой составляет композиция кодировщиков трансформера. BERT является автокодировщиком. В каждом слое кодировщика применяется двустороннее внимание, что позволяет модели учитывать контекст с обеих сторон от рассматриваемого токена, а значит, точнее определять значения токенов. Также мы решили поэксперементировать с моделью под названием XLNET. Это авторегрессионная модель (autoregressive language modeling, AR LM). Она пытается предсказать следующий токен по последовательности предыдущих. В классических авторегрессионных моделях эта контекстная последовательность берется независимо из двух направлений исходной строки. XLNet обобщает этот метод и формирует контекст из разных мест исходной последовательности. Как он это делает. Он берет все (в теории) возможные перестановки исходной последовательности и предсказывает каждый токен в последовательности по предыдущим.При этом контекст — это не мешок слов. Информация об изначальном порядке токенов также подается в модель. Если проводить аналогии с BERT-ом, то получается, мы не маскируем токены заранее, а как бы используем разные наборы скрытых токенов при разных перестановках. При этом исчезает и вторая проблема BERT-а — отсутствие скрытых токенов при использовании предобученной модели. В случае XLNet на вход поступает уже вся последовательность, без масок.

3.2 Результаты

В связи с ограниченными вычислительными ресурсами нашего окружения: мы будем обучать модели на 5 эпохах. В качестве метрик качества будем использовать f1 меру, также будем смотреть на accuracy и balanced accuracy. Посмотрим на результаты различных архитектур:

- BERT:

```
Epoch 1
Training loss: 1.0053112776577473
Validation loss: 0.9743953788280487
Accuracy Score: 0.51375
Balanced Accuracy Score: 0.45834515905898127
F1 Score (Macro): 0.4657391405562766

Epoch 2
Training loss: 0.8657778690755368
Validation loss: 0.8253409966826439
Accuracy Score: 0.6525
Balanced Accuracy Score: 0.6085899823732399
F1 Score (Macro): 0.6261238889000911

Epoch 3
Training loss: 0.6798073357716202
Validation loss: 0.8260481537878513
Accuracy Score: 0.66375
Balanced Accuracy Score: 0.630553742169576
F1 Score (Macro): 0.6495111535814925

Epoch 4
Training loss: 0.5189956092834472
Validation loss: 0.8835395184159279
Accuracy Score: 0.6625
Balanced Accuracy Score: 0.6152646834022564
F1 Score (Macro): 0.6357058637021183

Epoch 5
Training loss: 0.4070987550728023
Validation loss: 0.8977273650467396
Accuracy Score: 0.67
Balanced Accuracy Score: 0.6295319000705114
F1 Score (Macro): 0.6482920916540699
```

Видим что качество заметно лучше, чем у fastext модели, но все равно до baseline 2 мы пока не дотягиваем.

- RoBERTa:

```
Epoch 1
Training loss: 1.0009239083528518
Validation loss: 0.889594258069992
Accuracy Score: 0.555
Balanced Accuracy Score: 0.5222700077339922
F1 Score (Macro): 0.5400866049109837

Epoch 2
Training loss: 0.8111034320294856
Validation loss: 0.7389097203314304
Accuracy Score: 0.68875
Balanced Accuracy Score: 0.6640746015827131
F1 Score (Macro): 0.6805178616627053

Epoch 3
Training loss: 0.6383064908534288
Validation loss: 0.6844336014986038
Accuracy Score: 0.73125
Balanced Accuracy Score: 0.7130132251482024
F1 Score (Macro): 0.7274913220773094

Epoch 4
Training loss: 0.5025479396246374
Validation loss: 0.7297434360533953
Accuracy Score: 0.7225
Balanced Accuracy Score: 0.7068617949214965
F1 Score (Macro): 0.7206154611748176

Epoch 5
Training loss: 0.4026647930685431
Validation loss: 0.769295623600483
Accuracy Score: 0.72625
Balanced Accuracy Score: 0.7102887594775978
F1 Score (Macro): 0.7249823974057938
```

Здесь уже качество выросло начиная с 3 эпохи качество заметно увеличилось.

- DeBERTa:

```
Epoch 1
Training loss: 0.9034002756141126
Validation loss: 0.7165155307203531
Accuracy Score: 0.6975
F1 Score (Macro): 0.6942833837756966

Epoch 2
Training loss: 0.69733188922517
Validation loss: 0.7045273529924452
Accuracy Score: 0.74625
F1 Score (Macro): 0.7426864245649826

Epoch 3
Training loss: 0.5549221717589535
Validation loss: 0.9382986612711102
Accuracy Score: 0.74
F1 Score (Macro): 0.7401551217331948

Epoch 4
Training loss: 0.42052237377647544
Validation loss: 1.1248413733334746
Accuracy Score: 0.74
F1 Score (Macro): 0.7358655253494718

Epoch 5
Training loss: 0.30757009107648625
Validation loss: 1.2626893708182616
Accuracy Score: 0.7475
F1 Score (Macro): 0.7472400828864125
```

Тут заметно, что качество уже лучше чем у предыдущих моделей + мы побили baseline2!

- ALBERT:

```
Epoch 1
Training loss: 0.9325758807361126
Validation loss: 0.7961407992243766
Accuracy Score: 0.64625
Balanced Accuracy Score: 0.6144541160764393
F1 Score (Macro): 0.6331428409648462

Epoch 2
Training loss: 0.7201566229388118
Validation loss: 0.742730643749237
Accuracy Score: 0.6775
Balanced Accuracy Score: 0.6308812321141977
F1 Score (Macro): 0.6537940238788569

Epoch 3
Training loss: 0.49826346583664416
Validation loss: 0.8002311588823795
Accuracy Score: 0.68125
Balanced Accuracy Score: 0.6524196115370677
F1 Score (Macro): 0.6676126430480925

Epoch 4
Training loss: 0.2982128457399085
Validation loss: 0.8928102475404739
Accuracy Score: 0.69375
Balanced Accuracy Score: 0.6628519220342711
F1 Score (Macro): 0.6789595015948109

Epoch 5
Training loss: 0.17323393737897277
Validation loss: 0.9697390080615879
Accuracy Score: 0.67875
Balanced Accuracy Score: 0.6480789813690527
F1 Score (Macro): 0.6646706538656311
```

Здесь у нас качество не сильно улучшилось по сравнению с BERT моделью.

- DistilBERT:

```
Epoch 1
Training loss: 1.0122031019628048
Validation loss: 0.9795107293128967
Accuracy Score: 0.49375
Balanced Accuracy Score: 0.5123467274667793
F1 Score (Macro): 0.4661084685904428

Epoch 2
Training loss: 0.8968029316514731
Validation loss: 0.8655526888370514
Accuracy Score: 0.59
Balanced Accuracy Score: 0.5454594075619384
F1 Score (Macro): 0.5685783223706081

Epoch 3
Training loss: 0.7891295550763607
Validation loss: 0.8394150179624558
Accuracy Score: 0.62
Balanced Accuracy Score: 0.5860443471734841
F1 Score (Macro): 0.6016926668197896

Epoch 4
Training loss: 0.6742814549058675
Validation loss: 0.8389324712753295
Accuracy Score: 0.62375
Balanced Accuracy Score: 0.5884268446241191
F1 Score (Macro): 0.6054177005212962

Epoch 5
Training loss: 0.5824804983660579
Validation loss: 0.85509545981884
Accuracy Score: 0.62875
Balanced Accuracy Score: 0.5896948174040971
F1 Score (Macro): 0.6093214446762051
```

Качество тоже не подходящее для нас.

- XLNET:

```
Epoch 1
Training loss: 1.024374841451645
Validation loss: 1.0612417185306549
Accuracy Score: 0.45625
Balanced Accuracy Score: 0.3836532059763727
F1 Score (Macro): 0.3050568150328588

Epoch 2
Training loss: 0.9261543945223093
Validation loss: 0.9308994615077972
Accuracy Score: 0.54125
Balanced Accuracy Score: 0.499380221768281 45
F1 Score (Macro): 0.49969268353689555

Epoch 3
Training loss: 0.8297348654270172
Validation loss: 0.8717935499548912
Accuracy Score: 0.61625
Balanced Accuracy Score: 0.5960678084623575
F1 Score (Macro): 0.6049046595306939

Epoch 4
Training loss: 0.7011746856570243
Validation loss: 0.883614002764225
Accuracy Score: 0.65375
Balanced Accuracy Score: 0.6203336140390001
F1 Score (Macro): 0.6374154811515863

Epoch 5
Training loss: 0.5933672348409891
Validation loss: 0.8753544819355011
Accuracy Score: 0.675
Balanced Accuracy Score: 0.6395893096736706
F1 Score (Macro): 0.6578503069441264
```

К сожалению авторегриссионная модель не показала качество лучше, чем DeBERTa. У нас есть предположение, что это связано с тем что эта модель очень затратна на ресурсы для обучения и для выбивания лучшего качества

нам следовало бы затратить больше ресурсов, чем те которыми мы располагаем.

Не забудем про ablation study, в ходе обработки данных мы использовали склеинное название кликбейта и его текст, теперь попробуем убрать это склеивание и посмотрим как у нас измениться качество модели DeBERTa:

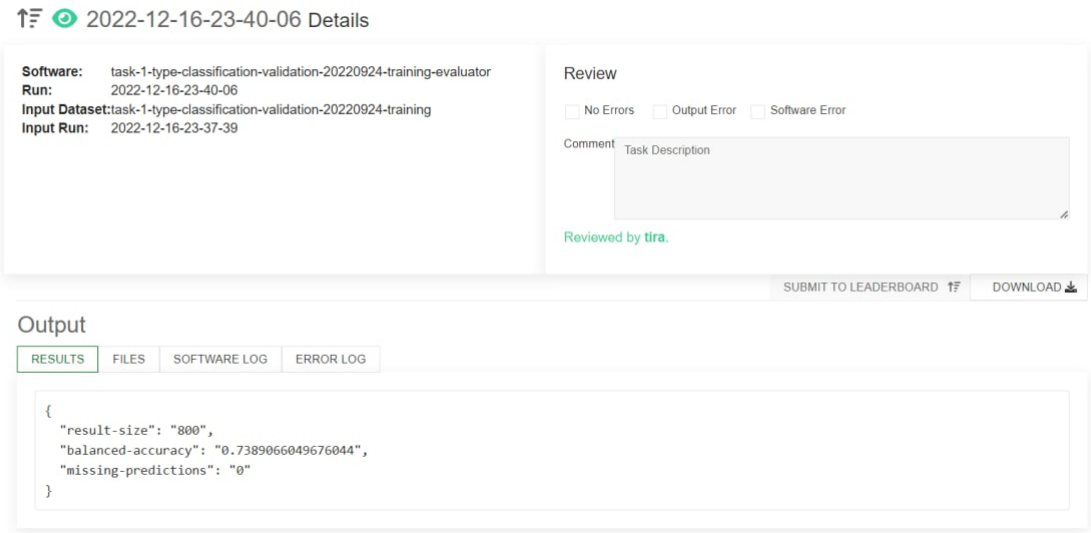
```
Some weights of the model checkpoint at microsoft/deberta-base were not used when initi
- This IS expected if you are initializing DebertaForSequenceClassification from the ch
- This IS NOT expected if you are initializing DebertaForSequenceClassification from th
Some weights of DebertaForSequenceClassification were not initialized from the model ch
You should probably TRAIN this model on a down-stream task to be able to use it for pre
epochs: 3
f1: 0.6667332980634951
balanced accuracy: 0.6443346184423406

Some weights of the model checkpoint at microsoft/deberta-base were not used when initi
- This IS expected if you are initializing DebertaForSequenceClassification from the ch
- This IS NOT expected if you are initializing DebertaForSequenceClassification from th
Some weights of DebertaForSequenceClassification were not initialized from the model ch
You should probably TRAIN this model on a down-stream task to be able to use it for pre
epochs: 4
f1: 0.6805858856479127
balanced accuracy: 0.6583886371037572

Some weights of the model checkpoint at microsoft/deberta-base were not used when initi
- This IS expected if you are initializing DebertaForSequenceClassification from the ch
- This IS NOT expected if you are initializing DebertaForSequenceClassification from th
Some weights of DebertaForSequenceClassification were not initialized from the model ch
You should probably TRAIN this model on a down-stream task to be able to use it for pre
epochs: 5
f1: 0.6657503337921488
balanced accuracy: 0.6418172353279427
```

Видим, что у нас качество заметно ухудшилось. Значит название кликбейта очень сильно влияет на наше предсказание.

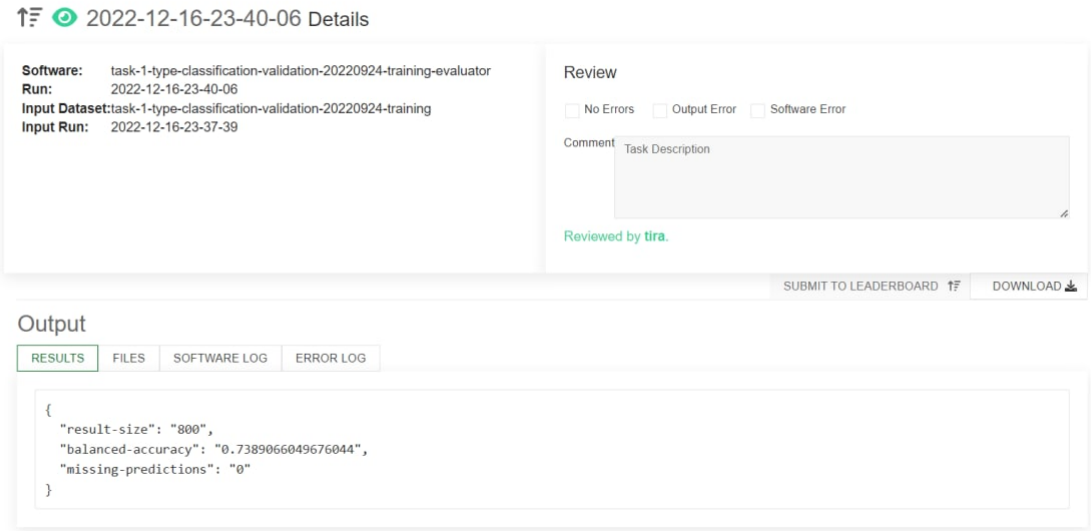
На основе проведенных экспериментов мы решили остановиться на модели DeBERTa: попробуем заняться улучшение гиперпараметров, тут мы ограничены ресурсами нашего окружения по-этому особо разгуляться не сможем, однако чуть-чуть улучшить качество нам все же удалось:

```
Epoch 1
Training loss: 0.901367158703506
Validation loss: 0.7222601614892483
Accuracy Score: 0.71375
Balanced Accuracy Score: 0.6735787898020216
F1 Score (Macro): 0.6981170476838537

Epoch 2
Training loss: 0.6959486353117973
Validation loss: 0.7107981157302856
Accuracy Score: 0.7175
Balanced Accuracy Score: 0.6906271751177657
F1 Score (Macro): 0.7070199160479178

Epoch 3
Training loss: 0.5709038489812519
Validation loss: 0.8212458357121796
Accuracy Score: 0.76125
Balanced Accuracy Score: 0.7354373292725013
F1 Score (Macro): 0.7521244015823233

Epoch 4
Training loss: 0.4457311793687404
Validation loss: 1.1630821202800143
Accuracy Score: 0.74
Balanced Accuracy Score: 0.7279942640098382
F1 Score (Macro): 0.7314961380556236

Epoch 5
Training loss: 0.3512469612697896
Validation loss: 1.2593289360788185
Accuracy Score: 0.74875
Balanced Accuracy Score: 0.7411170918633605
F1 Score (Macro): 0.7444436659148596
```

Так как организаторы соревнования до сих пор не предоставили нам доступ к тестовой выборке данных, мы сделаем сабмит по валидационному набору данных, к счастью такая возможность на сайте имеется.

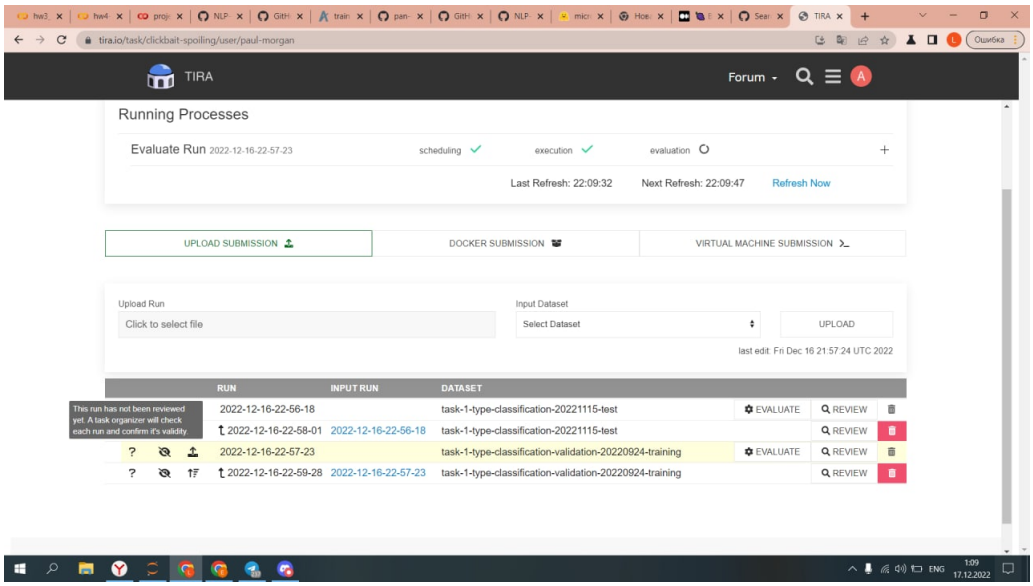


Далее мы решили сделать сабмит и решения авторов, чтобы посмотреть на baseline на balanced accuracy метрике:



Как видим наша модель побила baseline2.

Далее нам стоит ждать подтверждения сабмита от авторов соревнования + момент когда нам откроют доступ к тестовым данным. К сожалению авторы до сих пор не просмотрели наш сабмит и не предоставили доступ к тестовым данным.



Соответственно в связи с этим результаты нашего сабмита не отображаются в leaderboarde.

3.3 Итоги

Мы провели огромную работу по решению задачи классификации типа кликбэйта на основе его описания. Исследовали базовую архитектуру такую как fasttext. Поняли, что для решения задачи нужна более сложная и мощная архитектура. Изучили устройство архитектуры современных трансформеров. Сумели также поэксперементировать с различными вариациями и модификациями архитектуры BERT. Также затронули авторегриссионную модель XLNET. Самое лучшее качество продемонстрировала модель DeBERTa, в принципе что и логично, так как в статье которую мы использовали в качестве камертона, авторы тоже выявили что архитектура модели DeBERTa проявила себя лучше всего. К сожалению, чтобы сравниться с остальными участниками соревнования у нас не получилось сделать сабмит на тестовых данных, в связи с отсутствием этих данных.