

Пояснительная записка

к дипломному проекту на тему:

**"**Определение жанровых тегов фильмов по их краткому описанию**"**

**Автор:** Юрий Ефремов

**Группа:** DSU-35

**Ментор:** Татьяна Булгакова

**Презентация:**

**Репозиторий:**

Оглавление

[1. Постановка задачи 4](#_Toc168264369)

[1.1. АКТУАЛЬНОСТЬ 4](#_Toc168264370)

[1.2. ПРОБЛЕМА 4](#_Toc168264371)

[1.3. ЦЕЛЬ РАБОТЫ 4](#_Toc168264372)

[1.4. ЗАДАЧА 4](#_Toc168264373)

[1.5. МЕТРИКИ 5](#_Toc168264374)

[1.6. МЕТОДИКА ОЦЕНКИ 6](#_Toc168264375)

[2. Получение и предобработка данных 8](#_Toc168264376)

[2.1. ПОЛУЧЕНИЕ ДАННЫХ 8](#_Toc168264377)

[2.2. ПРЕДОБРАБОТКА ДАННЫХ 9](#_Toc168264378)

[3. Анализ данных 10](#_Toc168264379)

[3.1. РАСПРЕДЕЛЕНИЕ ФИЛЬМОВ ПО ЖАНРАМ 10](#_Toc168264380)

[3.2. КОРРЕЛЯЦИЯ МЕЖДУ ЖАНРАМИ 11](#_Toc168264381)

[3.3. РАСПРЕДЕЛЕНИЕ ЖАНРОВ ПО ГОДАМ 12](#_Toc168264382)

[3.4. РАСПРЕДЕЛЕНИЕ ЖАНРОВ ПО СТРАНАМ 13](#_Toc168264383)

[4. Лемматизация текстов и разделение на выборки 14](#_Toc168264384)

[4.1. ЛЕММАТИЗАЦИЯ 14](#_Toc168264385)

[4.2. РЕЗУЛЬТАТ ЛЕММАТИЗАЦИИ 15](#_Toc168264386)

[4.3. Разделение на обучающую и тестовую выборки 15](#_Toc168264387)

[5. Моделирование. Классические подходы 16](#_Toc168264388)

[5.1. TFIDF & LOGISTIC REGRESSION 16](#_Toc168264389)

[5.1.1. Модель на униграммах 17](#_Toc168264390)

[5.1.2. Модель на униграммах и биграммах 18](#_Toc168264391)

[5.2. TFIDF & CATBOOST 20](#_Toc168264392)

[5.2.1. Модель на униграммах 21](#_Toc168264393)

[5.2.2. Модель на униграммах и биграммах 22](#_Toc168264394)

[6. Моделирование. Нейросетевые подходы 25](#_Toc168264395)

[6.1. МЕТОДИКА ОБУЧЕНИЯ 25](#_Toc168264396)

[6.1.1. Создание датасетов 25](#_Toc168264397)

[6.1.2. Функции для обучения и валидации 26](#_Toc168264398)

[6.1.3. Функции логирования и отображения метрик 27](#_Toc168264399)

[6.1.4. Инициализация и обучение 28](#_Toc168264400)

[6.1.5. Эксперименты 29](#_Toc168264401)

[6.2. ТРАНСФОРМЕР RUBERT TINY 29](#_Toc168264402)

[6.3. ТРАНСФОРМЕР RUBERT BASE 32](#_Toc168264403)

[6.4. ТРАНСФОРМЕР distilbert base ru 34](#_Toc168264404)

[6.5. ТРАНСФОРМЕР XLM-ROBERTA BASE 37](#_Toc168264405)

[7. Сравнение моделей 39](#_Toc168264406)

1. Постановка задачи
   1. АКТУАЛЬНОСТЬ

На сегодняшний день киноиндустрия является одной из самых значимых и динамично развивающихся отраслей в экономике развлечений. Ежегодно выпускаются тысячи новых фильмов, и с каждым годом эта цифра только росла. Благодаря развитию и распространению технологий, кино становится доступным для широкой аудитории по всему миру, что увеличивает разнообразие жанров и тематик, представленных в кинематографе. Появляются также и новые форматы потребления контента, различные онлайн-кинотеатры и медиа-платформы. В условиях такого многообразия, классификация и систематизация каталогов кино становится важной задачей как для создателей контента, так и для потребителей.

Ключевым аспектом в классификации фильмов является присвоение жанровых тегов. Такие теги помогают зрителям быстрее находить кино, соответствующее их интересам и предпочтениям. Однако процесс их присвоения часто требует значительных усилий и времени, особенно когда речь идет о большом количестве новых релизов. В этой связи автоматизация такого процесса с использованием методов машинного обучения приобретает особую значимость.

* 1. ПРОБЛЕМА

Существующие методы определения жанров могут базироваться на ручной классификации, что является трудоемким и субъективным процессом. С увеличением объема информации и количества выпускаемого кино необходимость в автоматизированных и объективных подходах становится все более очевидной. Краткие описания фильмов, которые часто публикуются вместе с релизами, содержат ключевую информацию о сюжете, темах и настроении фильма. Эти описания могут быть использованы в качестве основы для автоматического определения жанровых тегов, что существенно упростит и ускорит процесс классификации.

Дополнительным усложняющим фактором является то, что отдельно взятый фильм сложно отнести к какому-либо одному жанру. Часто в кино присутствуют элементы нескольких жанров, а сами создатели для привлечения большей аудитории осознанно создают мульти-жанровое кино.

* 1. ЦЕЛЬ РАБОТЫ

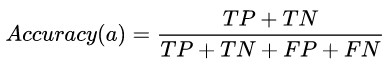
Целью этой работы является разработка, оценка и сравнение моделей машинного обучения для определения жанровых тегов фильмов на основе их кратких описаний. В рамках исследования будут рассмотрены классические и нейросетевые алгоритмы машинного обучения, их эффективность и точность в задаче multi-label классификации текстовых данных.

* 1. ЗАДАЧА

Получить достаточное количество данных, проанализировать их и подготовить к обучению моделей. Для решения задачи multi-label классификации обучить несколько классических алгоритмов и несколько трансформенных нейросетевых моделей. Оценить обученные модели и сравнить их показатели.

* 1. МЕТРИКИ

**Accuracy**

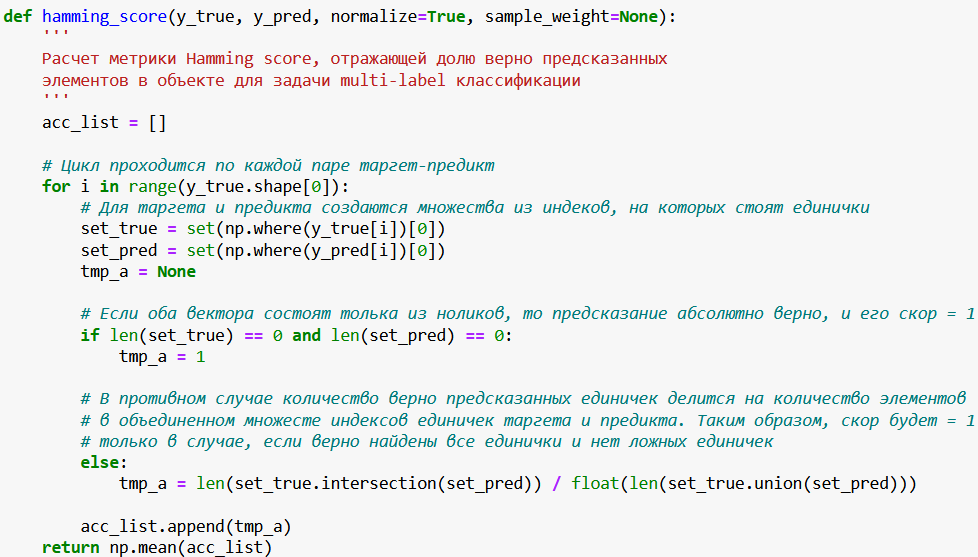


Доля правильно классифицированных объектов. В случае с multi-label классификацией один объект представляет из себя вектор из нолей и единичек, где единички соответствуют тегам, относящимся к фильмам. Объект будет считаться классифицированным верно, если предсказанный вектор полностью совпадает с таргетом.

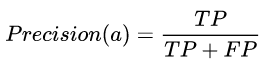
**Hamming score (Расстояние Хэмминга)**

Доля правильно классифицированных элементов в объекте. Для задачи multi-label классификации такая метрика подходит, пожалуй, даже лучше, чем accuracy, поскольку она будет учитывать те случаи, в которых модель частично верно предсказала объект. Для оценки модели считается среднее расстояние Хэмминга по всем объектам.

Для расчета метрики использовалась следующая функция[[1]](#footnote-1):



**Precision (Точность)**



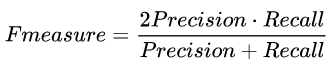
На уровне класса (жанра) точность показывает какая доля элементов этого класса среди всех элементов, отнесенных моделью ко классу по всем объектам, было распознано верно. Для оценки работы модели в целом будут рассматриваться усреднения точности по всем классам.

**Recall (Полнота)**



На уровне класса (жанра) полнота показывает какая доля элементов этого класса среди всех элементов, реально относящихся ко классу по всем объектам, было распознано моделью верно. Для оценки работы модели в целом будут рассматриваться усреднения полноты по всем классам.

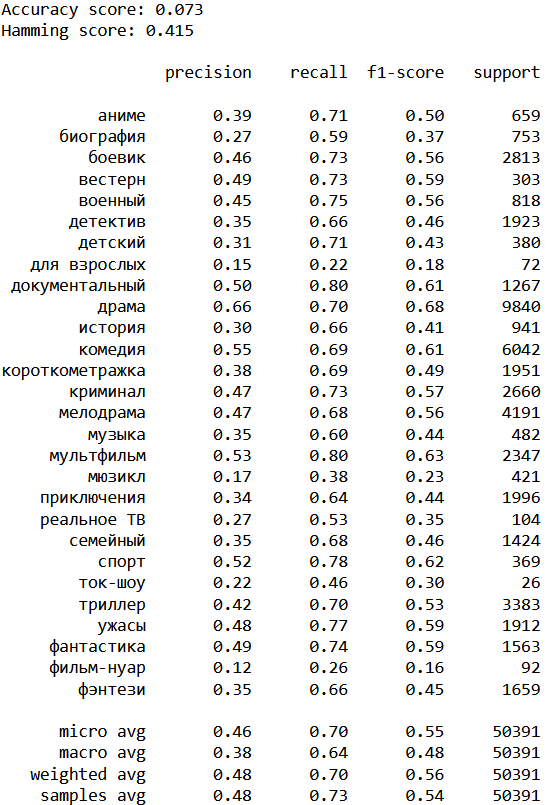
**F1 score (F1-мера)**



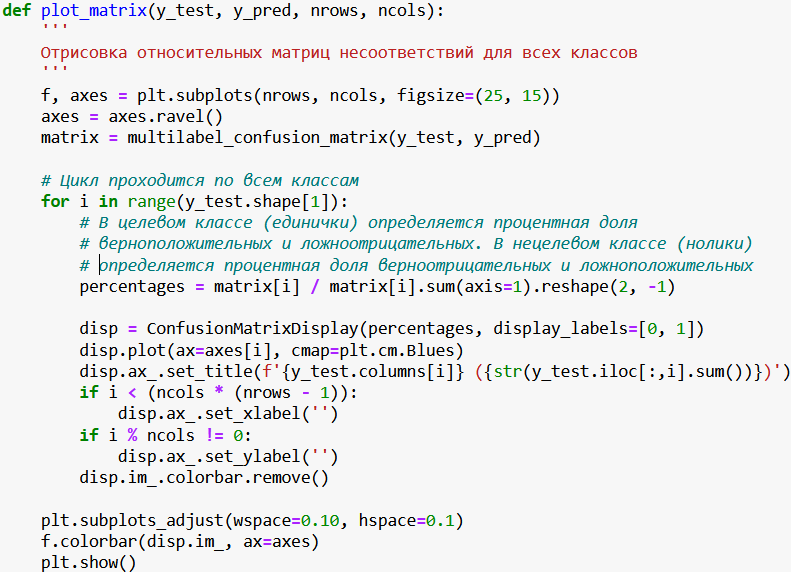
Среднее гармоническое точности и полноты. Для оценки работы модели в целом будут рассматриваться усреднения F1-меры по всем классам.

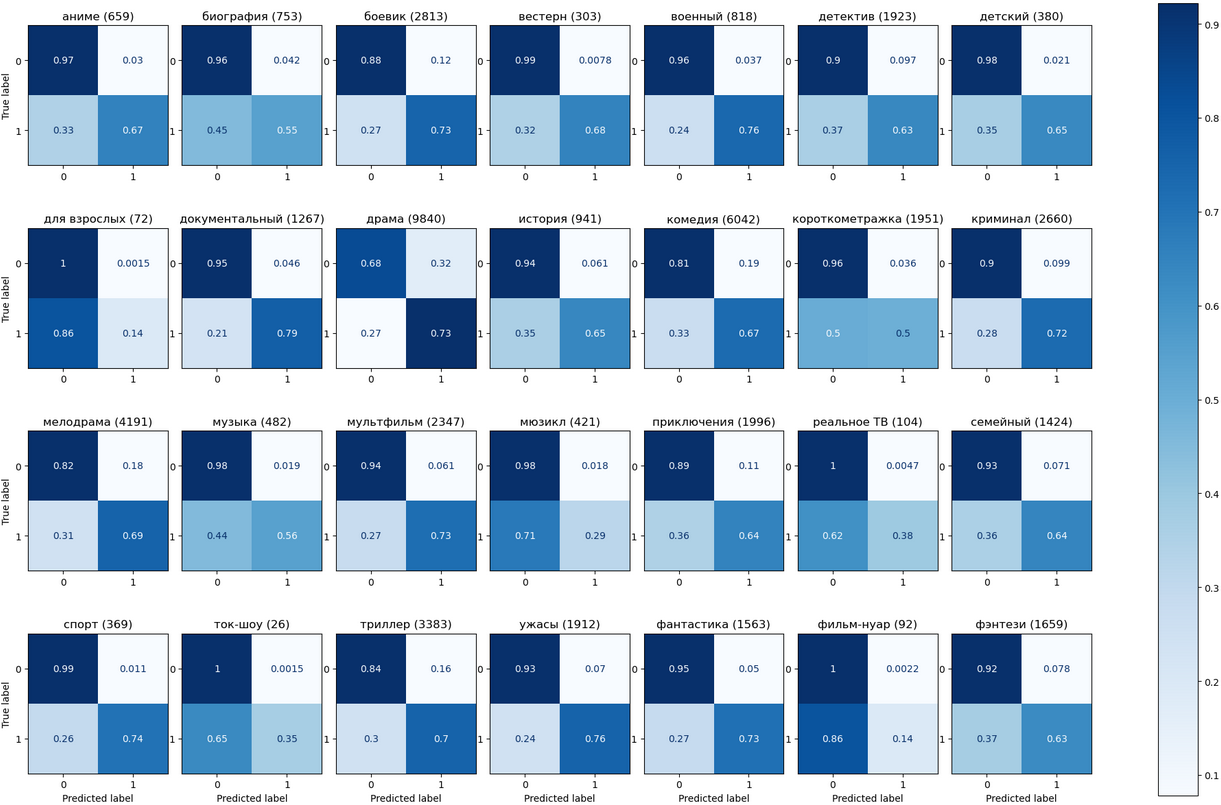
* 1. МЕТОДИКА ОЦЕНКИ

Для выведения метрик на экран использовались методы **accuracy\_score** и **classification\_report** из библиотеки **scikit-learn**, а также приведенная выше функция **hamming\_score**.



Кроме этого, для более визуально-направленной оценки была написана функция для вывода на экран матрицы несоответствий для каждого класса. При этом для большей наглядности абсолютные количественные значения в матрицах были заменены относительными.





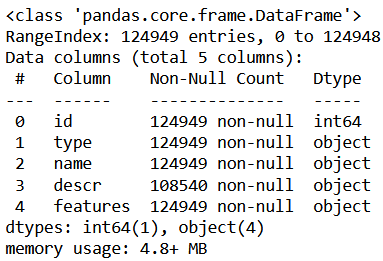
1. Получение и предобработка данных
   1. ПОЛУЧЕНИЕ ДАННЫХ

В качестве источника данных было принято решение использовать платформу Кинопоиск[[2]](#footnote-2), поскольку это крупнейший и наиболее популярный русскоязычный веб-сервис, посвященный сфере кино. На сайте содержится информация о почти 1 млн. фильмов, сериалов и телепередач.

Для автоматизированного получения данных с сайта использовался парсер[[3]](#footnote-3), который проходится по страницам каталога сайта, собирает ссылки на фильмы, затем проходится по страницам фильмов, добавляя информацию о них в датасет.



С помощью парсера было обработано 2500 страниц каталога Кинопоиска и получена информация о 125 тыс. фильмов.



* 1. ПРЕДОБРАБОТКА ДАННЫХ

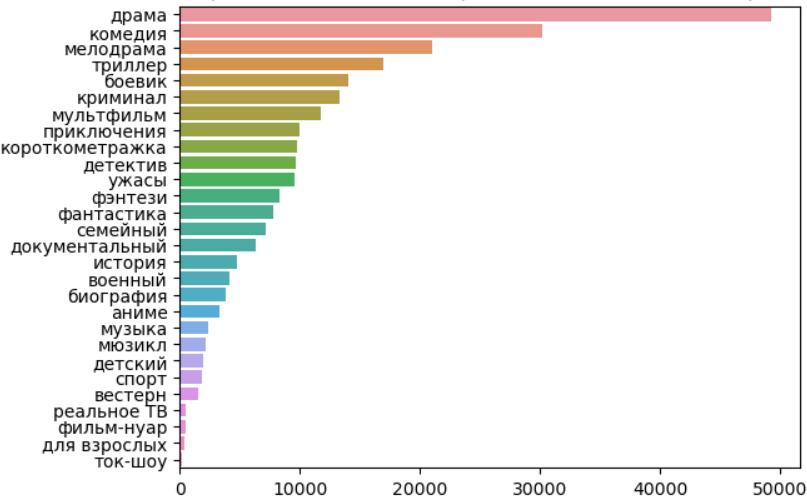
В ходе предобработки данных были проделаны следующие шаги:

* Удаление фильмов без описания, либо таких, в которых описание состоит только из пробельных символов
* Удаление дубликатов, и слишком коротких описаний
* Преобразование в словарь строкового представления словаря с параметрами фильмов, куда входят и жанры
* Создание колонки с жанрами и затем ее бинаризация
* Удаление "мусорных" и микро-жанров
* Удаление фильмов без жанров

После предобработки данных в датасете осталось **108049** фильмов, на которых в дальнейшем обучались и тестировались модели.



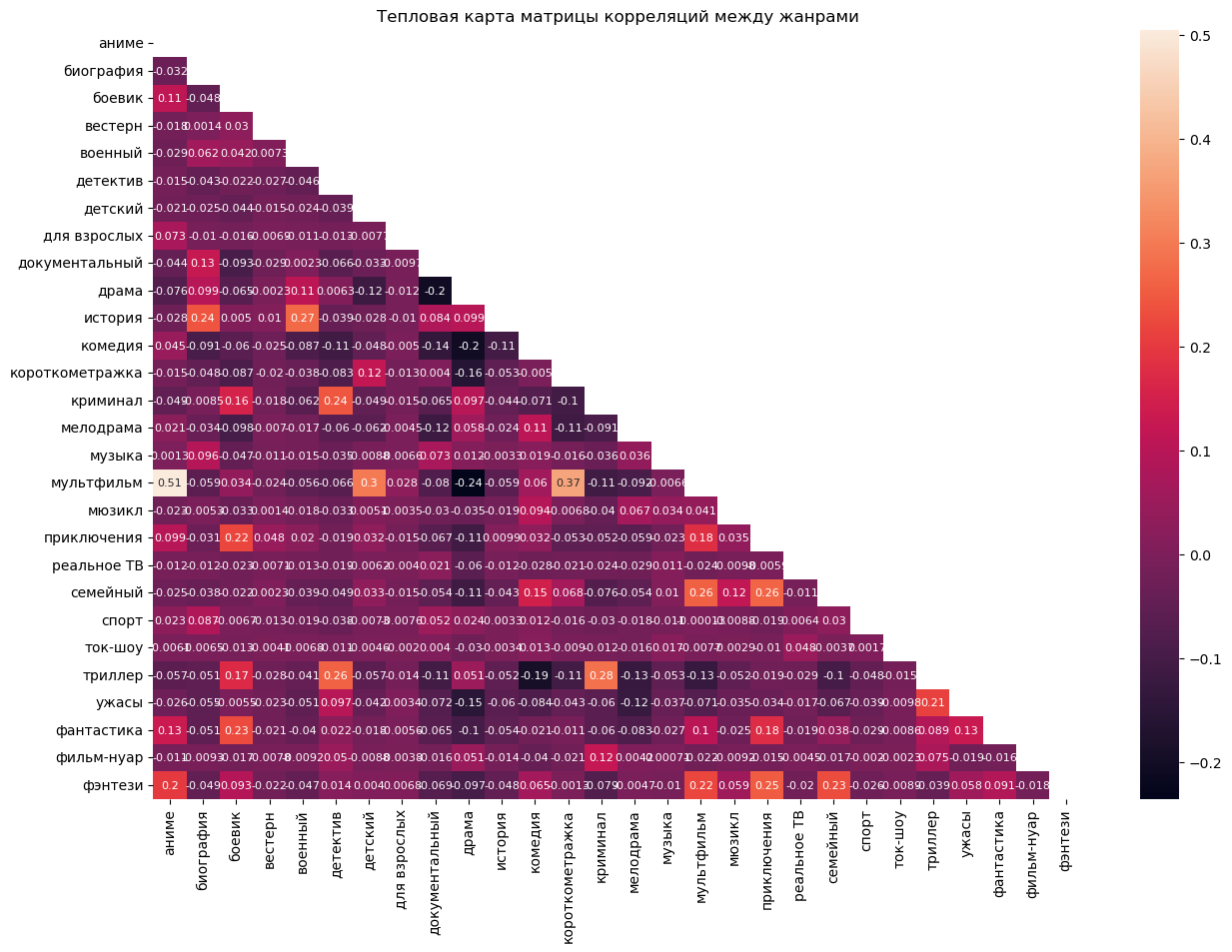
1. Анализ данных
   1. РАСПРЕДЕЛЕНИЕ ФИЛЬМОВ ПО ЖАНРАМ



Налицо сильный дисбаланс классов. Самые старейшие из жанров в искусстве **драма** и **комедия** по праву занимают первые две строчки. Почти половина всех фильмов имеют жанровый тег **драма**. При этом в датасете присутствуют и крайне мало представленные жанры, такие как **вестерн** и **фильм-нуар**, бывшие в свое время довольно популярными, но с годами уступившие дорогу более современным и универсальным жанрам **боевик**, **триллер** и **криминал**.

Не стоит забывать о том, что решается задача multi-label классификации и одному фильму может соответствовать несколько жанров. К примеру, в последнее время популярным стал тег **драмеди** – гибридный жанр, означающий комедийную драму. Давно уже снимаются **черные комедии**, совмещающие юмор с **триллером**, или **ужасами**. Жанры **история** и **военный** очень часто сочетаются в силу своей тематики. В нашем представлении рука об руку должны идти **боевики** и **приключения**, или **ужасы** и **фантастика**. В этой связи интересно посмотреть на то, какие жанры чаще всего друг с другом соседствуют.

* 1. КОРРЕЛЯЦИЯ МЕЖДУ ЖАНРАМИ



Несколько наблюдений можно сделать из построенной тепловой карты матрицы корреляций между жанрами. Наиболее часто соседствуют друг с другом жанры **мультфильм** и **аниме**, что представляется вполне логичным результатом. **Мультфильм** также сильно скоррелирован с жанрами **детский** и **короткометражка** - к этому аналогично вопросов нет. Как было отмечено ранее, **исторические** фильмы часто снимаются на **военную** тематику, но и с жанром **биография** их регулярное соседство не кажется случайным. **Детективное** кино по самой своей задумке должно относиться к жанрам **криминал** и **триллер**, что здесь и наблюдается.

Также стоит отметить и наличие отрицательной корреляции в некоторых случаях. Наиболее сильно это выражено между жанрами **мультфильм** и **драма**, что можно объяснить по определению детской направленностью мультиков, где нет места сложным, «взрослым» взаимоотношениям. Вопреки набирающему популярность гибриду **драмеди**, окончательно подружить **драму** с **комедией** пока не удается, и отрицательная корреляция между этими основополагающими жанрами дает здесь о себе знать.

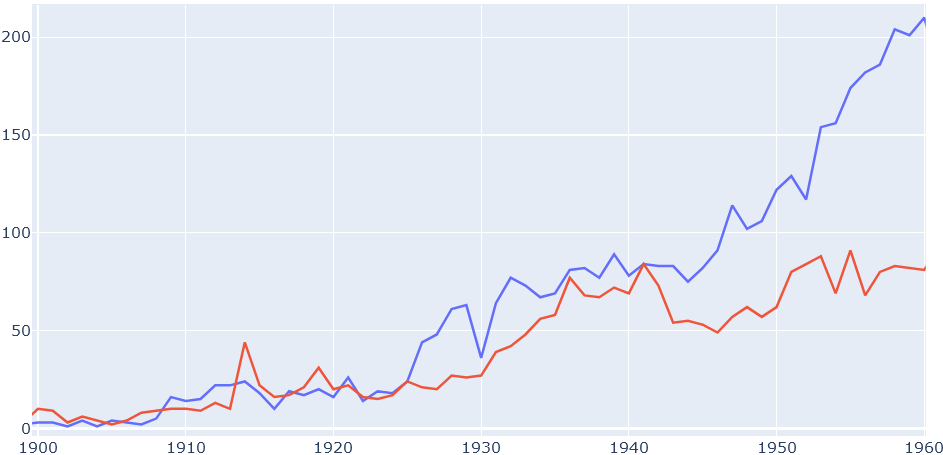
* 1. РАСПРЕДЕЛЕНИЕ ЖАНРОВ ПО ГОДАМ



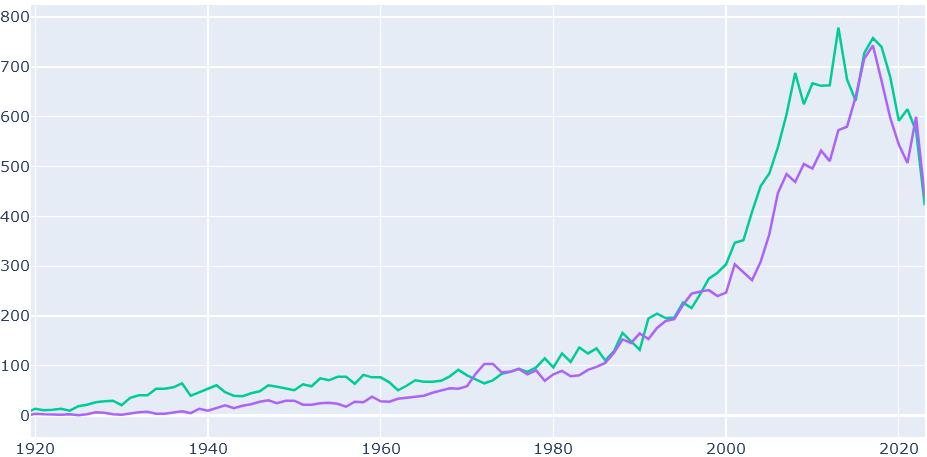
Для начала рассмотрю не всю историю кинематографа, а лишь отрезок, начиная с 1990 года, и не все жанры, а лишь топ-10 (**короткометражки** при этом я исключил, заменив на **ужасы**). За скобки также взяты будущий 2025 и настоящий 2024 годы. Вероятно также существует задержка в наполнении базы данных Кинопоиска, поэтому 2023 год можно считать не полным.

Вплоть до последних лет наблюдался поступательный, но устойчивый рост количества выпускаемых фильмов во всех основных жанрах. Не может не броситься в глаза некоторый спад, начавшийся после 2018 года, чему должны быть объяснением глобальная пандемия и последовавшая за ней напряженность в межгосударственных отношениях. Здесь необходимо отметить, что годом производства фильма на Кинопоиске является год его готовности, а не год выпуска в прокат или показа на кинофестивалях. Таким образом эффект задержки выхода фильмов на временной шкале проявляет себя раньше, чем наступление факторов напряжения в обществе.

Видно, что наиболее резко сократился выпуск **драм**, к 2023 году почти приблизившийся к выпуску **комедий**. Вероятно, ожидания людей от кино сместились в сторону более развлекательного контента, нежели серьезного, драматического. К слову, до 1941 года разрыв в выпуске **драм** и **комедий** не был таким разительным, и именно после 1941 года он начал расти.

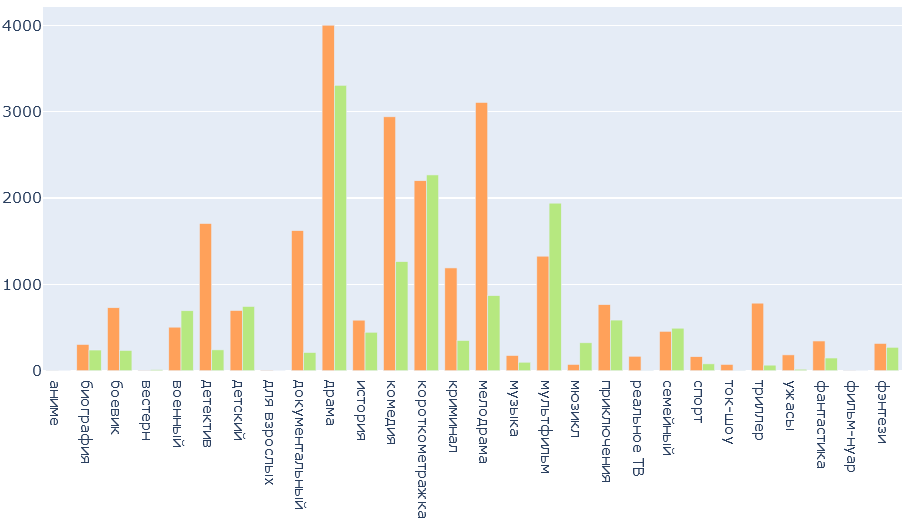


Интересно рассмотреть еще два весьма полярных жанра **триллер** (фиолетовый) и **мелодрама** (зеленый). Наибольшая отрицательная корреляция у **мелодрам** именно с **триллерами**. Вплоть до начала 70-х годов фильмов о любви стабильно выходило больше, чем фильмов с упором на саспенс. После этого и примерно до конца тысячелетия между жанрами наблюдается паритет. Предположу, что «бум» 70-х годов на **триллеры** связан с тем, что оформился и стал набирать популярность жанр полицейского триллера, а также возникла итальянская волна хорроров. В новом тысячелетии вкус кинозрителей стал снова тяготеть к **мелодрамам**, однако с появлением нескольких успешных серий фильмов о различных проклятиях и призраках, саспенс снова вошел в моду, и между жанрами опять установился паритет.



* 1. РАСПРЕДЕЛЕНИЕ ЖАНРОВ ПО СТРАНАМ

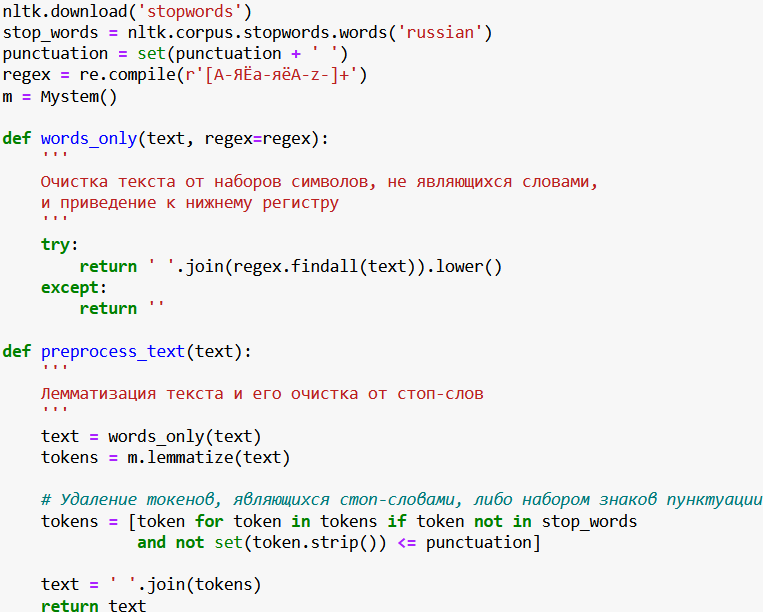
В этом разрезе я рассмотрю, как изменилось жанровое наполнение в отечественном кинематографе. Оранжевыми столбиками на графике представлена Россия, а зелеными СССР. Очевидно большее разнообразие жанров в российском кино, что должно быть объясняется большей современностью этой эпохи. Появились и **триллеры**, и **ужасы**, и **боевиков** с **детективами** стало больше. При этом **драмы** стали снимать относительно реже, чем **комедии**. **Комедий** стало в два раза больше, а **драм** лишь на 15-20%.



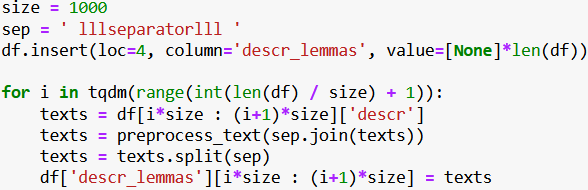
За исключением некоторых жанров, фильмов в России стало больше. Можно вспомнить о том насколько более сложной была система цезуры в советское время, и в конечном счете кино было именно государственным, идеологизированным продуктом. В России же включаются уже рыночные механизмы и в большей степени кино все-таки являлось продуктом частных студий. Тем интереснее обратить внимание на те жанры, в которых советское кино преуспело больше российского. Например, в СССР снималось чуть больше **детского** и **семейного** кино, ощутимо больше **военного** и **мюзиклов**. Но самым существенным мне видится разрыв в количестве **мультфильмов**. В наше время их стало на 30% меньше. Лишь отчасти я могу объяснить это тем, что современная мультипликация в России тяготеет больше к производству мультсериалов.

1. Лемматизация текстов и разделение на выборки
   1. ЛЕММАТИЗАЦИЯ

Для лемматизации текстов использовалась библиотека **PyMystem3**, а стоп-слова брались из библиотеки **NLTK**.



Известны проблемы со скоростью работы **PyMystem3**[[4]](#footnote-4), поэтому для ускорения было принято решение перед лемматизацией объединять тексты в группы по 1000 штук, и затем разъединять их обратно. При этом важно использовать разделитель. Таким образом лемматизация моего корпуса занимает лишь 6 минут.



* 1. РЕЗУЛЬТАТ ЛЕММАТИЗАЦИИ

ИСХОДНЫЙ ТЕКСТ:

Два охотника из коренных австралийских народов вынуждены противостоять армии вампиров.

ЛЕММАТИЗИРОВАННЫЙ ТЕКСТ:

охотник коренной австралийский народ вынужденный противостоять армия вампир

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ИСХОДНЫЙ ТЕКСТ:

Всё, что имеет значение для человека Сергея Довлатова, так или иначе написано писателем Сергеем Довлатовым. Из фрагментов произведений, писем, интервью, редакторских колонок составлен рассказ о живом человеке.

ЛЕММАТИЗИРОВАННЫЙ ТЕКСТ:

иметь значение человек сергей довлатов иначе написать писатель сергей довлатов фрагмент произведение письмо интервью редакторский колонка составлять рассказ живой человек

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ИСХОДНЫЙ ТЕКСТ:

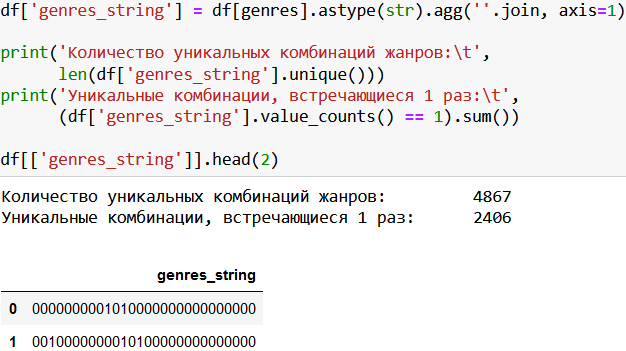
Это пять разных историй о пяти совершенно непохожих друг на друга женщинах. Единственное, что их объединяет, — искреннее желание обрести счастье, любовь и мать ее — нежность.

ЛЕММАТИЗИРОВАННЫЙ ТЕКСТ:

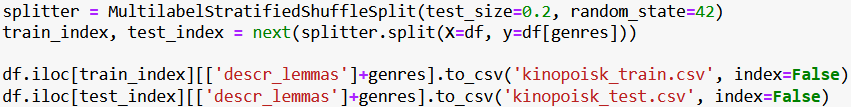
это пять разный история пять совершенно непохожий друг друг женщина единственный объединять искренний желание обретать счастие любовь мать нежность

* 1. Разделение на обучающую и тестовую выборки

Разделяя датасет на обучающую и тестовую выборки, важно помнить об имеющемся сильном дисбалансе классов. Если говорить о таргетах, являющихся векторами из нолей и единичек, то в датасете встречается 4867 таких уникальных комбинаций жанров. При этом из них 2406 комбинаций встречается только один раз. В такой ситуации остро встает вопрос о том, как правильно стратифицироваться по такому таргету.

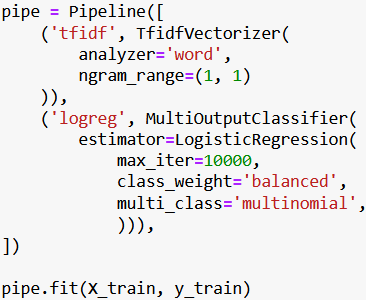


На помощь приходит библиотека **iterative-stratification** и встроенный в нее класс **MultilabelStratifiedShuffleSplit**[[5]](#footnote-5). С помощью него достигается пропорциональная представленность каждого класса (жанра) в обучающей и тестовой выборке. Для теста было оставлено 20% объектов, то есть **21536** фильмов. Соответственно в обучающую выборку попало **86513** фильмов.



1. Моделирование. Классические подходы
   1. TFIDF & LOGISTIC REGRESSION

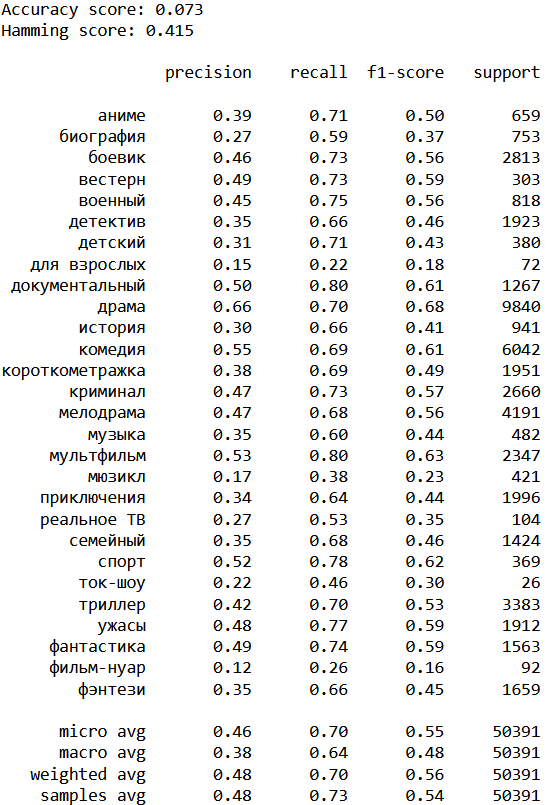
В качестве первого классического подхода к решению задачи была выбрана логистическая регрессия, построенная на значениях TF-IDF частотностей слов. Использовался класс **LogisticRegression** из библиотеки **scikit-learn**. При этом для случая multi-label классификации из этой же библиотеки над регрессией задействовалась обертка **MultiOutputClassifier**. Оттуда же был взят и векторизатор **TfidfVectorizer**.



Для сравнения было обучено две модели. Для первой TF-IDF векторизация производилась только на униграммах, а для второй на униграммах и биграммах. В ходе экспериментов была попытка задействовать также и триграммы, но время обучения бустинговых моделей (о которых будет рассказано в дальнейшем) на словаре такого размера было бы чрезмерно велико, поэтому от триграмм пришлось отказаться. Дополнительно отмечу, что ощутимого прироста в качестве при включении триграмм для логистической регрессии не было. Также на регрессии была опробована посимвольная TF-IDF векторизация в различных вариантах в диапазоне 3-7 символов, что тоже не дало каких-либо преимуществ в сравнении с 1-2-граммами.

* + 1. Модель на униграммах

Рассмотрим на тестовой выборке качество модели, обученной только на униграммах.



По крупнейшим жанрам уже с помощью этого метода достигаются весьма приличные показатели **precision** и **recall**. К примеру, для **драмы** это 0.66 и 0.70, а для **мультфильма** 0.53 и 0.80. Но и для микро-жанров ситуация не выглядит провальной. Например, **фильм-нуар** (92 штуки в тесте) набрал 0.12 и 0.26, а **ток-шоу** (26 штук в тесте) 0.22 и 0.46. **Accuracy score** при этом ожидаемо невысокий, всего 0.073, однако **hamming score**, на который стоит обращать большее внимание при решении моей задачи достиг очень хорошего значения 0.415.

Здесь стоит заметить, что при слепом угадывании вектора из нолей и единичек длиной 28, ожидаемый **accuracy score** был бы 2-28, что почти в 20 000 раз меньше, чем **accuracy score** первой модели 0.073.

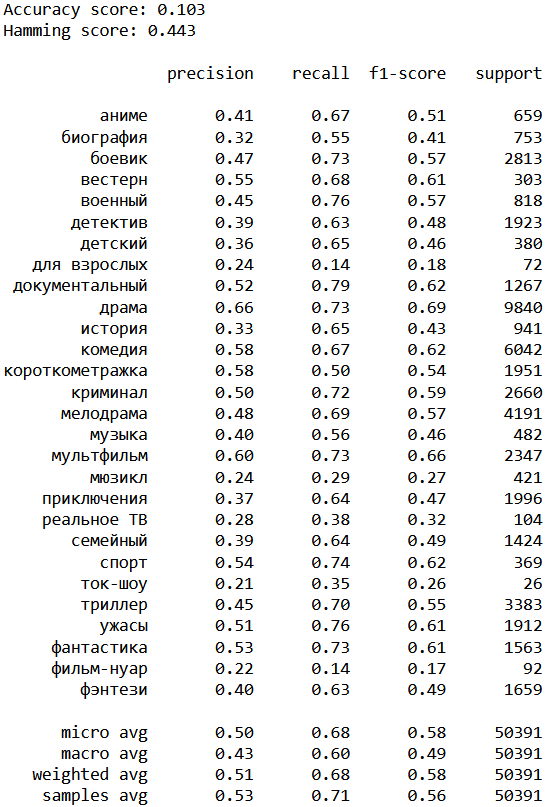
В среднем уклон модели наблюдается больше в сторону **recall**, на микро-усреднении он достиг 0.70, а **precision** получился 0.46. **F1 score** имеем 0.55.



* + 1. Модель на униграммах и биграммах

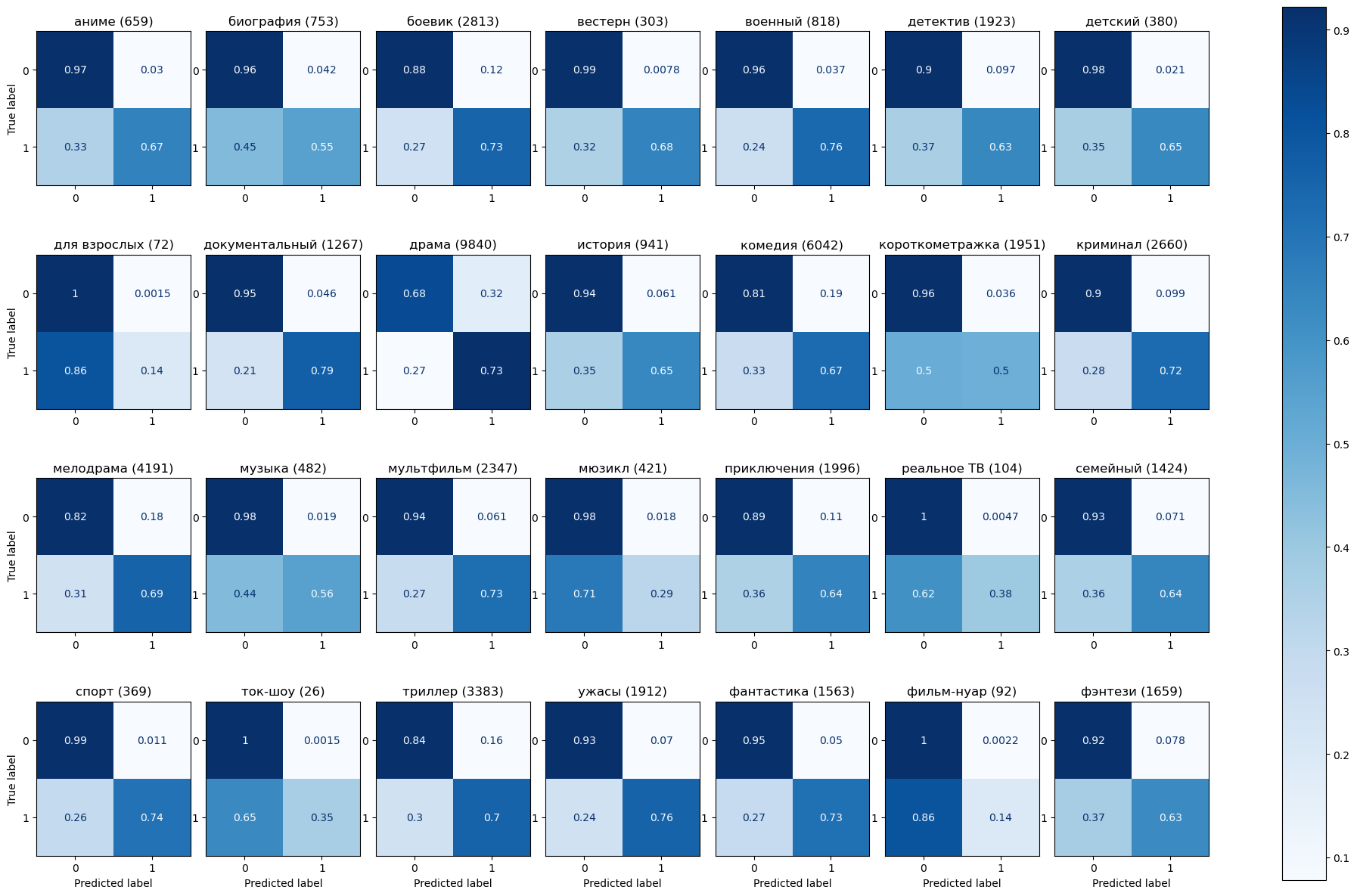
Необходимо уточнить, что логистическая регрессия на униграммах и биграммах обучалась не слишком долго. Было установлено максимальное количество итераций 10000 штук, что заняло 21 минуту на моем шестиядерном AMD Ryzen 5 5600H. Для сравнения, обучение на униграммах заняло всего 2 минуты.

Посмотрим удалось ли модели, обученной на униграммах и биграммах улучшить метрики на тестовой выборке.



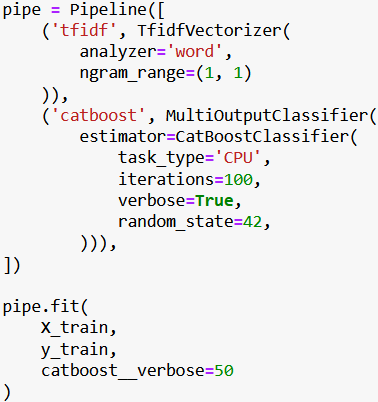
Заметен ощутимый прирост почти по всем основным метрикам. **Accuracy score** вырос с 0.073 до 0.103, **hamming score** с 0.415 до 0.443. **F1 score** в микро-усреднении был 0.55, а стал 0.58. **Precision** подрос с 0.46 до 0.5. Хотя произошло это в ущерб **recall**, который снизился с 0.7 до 0.68, все же модель стала разбираться в кино немного лучше своего предшественника.

Снижение **recall** произошло в большей степени за счет небольших жанров, в **ток-шоу** оно упало с 0.46 до 0.35, у **реального ТВ** с 0.53 до 0.38. Единственным крупным жанром, серьезно потерявшим очки **recall** стала **короткометражка**, где он был 0.69, а стал 0.5. В пользу такого вывода говорит и то, что **recall** в макро-усреднении упал сильнее, чем в микро-усреднении с 0.64 до 0.60.



* 1. TFIDF & CATBOOST

В качестве второго классического подхода была взята библиотека градиентного бустинга на деревьях решений **CatBoost**, векторизатором выступил тот же **TfidfVectorizer** из **scikit-learn**. Также был задействован и уже проявивший себя **MultiOutputClassifier**.

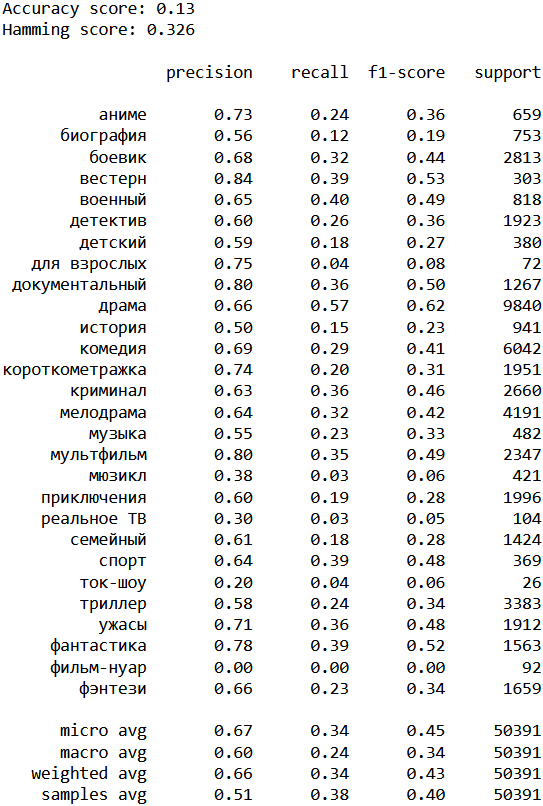


Аналогичным с логистической регрессией образом обучалось две модели, одна только на униграммах и вторая на униграммах и биграммах. Была также попытка задействовать встроенный в **CatBoost** функционал для multi-label классификации, но в ходе экспериментов он показал себя хуже, чем спайка **CatBoostClassifier** и **MultiOutputClassifier**. Бустинг от Яндекса хорошо самостоятельно подстраивает гиперпараметры под данные, поэтому со своей стороны я установил только количество итераций равным 100. Learning rate при этом установился 0.5.

* + 1. Модель на униграммах

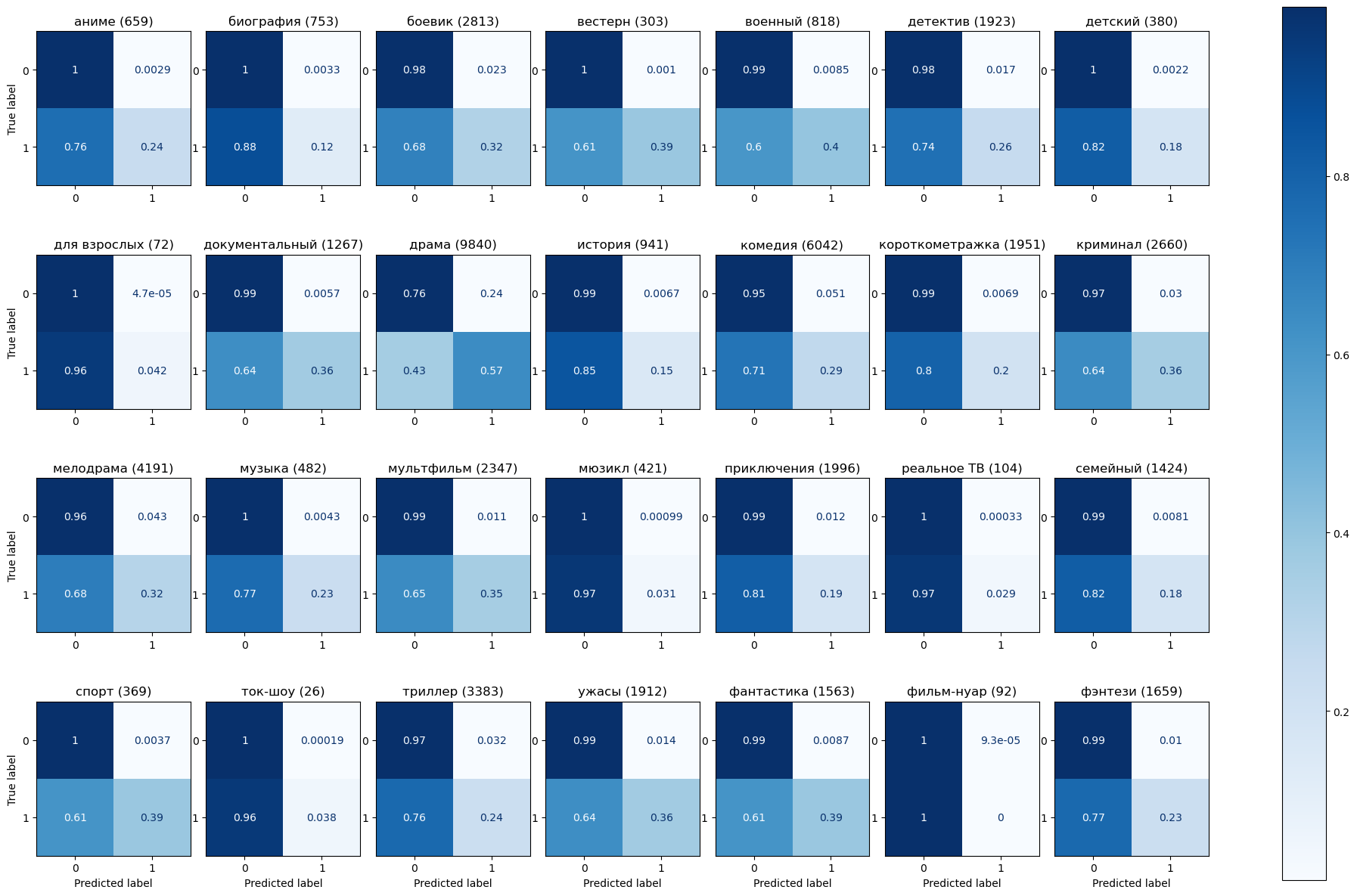
На обучение бустинга на униграммах ушло в 20 раз больше времени, чем на регрессию, а именно 41 минута. Обучение проводилось на том же процессоре AMD Ryzen 5 5600H. Хотя обучение на GPU было бы быстрее, было принято решение отказаться от обучения на видеокарте на Kaggle, поскольку в память GPU не вмещался датасет из униграмм + биграмм, а для корректного сравнения двух классических подходов хотелось обучить бустинг и на 1+2-граммах тоже.

Рассмотрим на тестовой выборке качество модели, обученной только на униграммах.



Несмотря на то, что некоторые метрики подросли, я оцениваю качество бустинговой модели ниже, чем работу логистической регрессии. Сравним между собой основные показатели для двух моделей, обученных на униграммах. **Accuracy score** подрос с 0.073 до 0.13, то есть на 78% стало больше абсолютно верно предсказанных таргетов. Несмотря на это **hamming score** снизился с 0.415 до 0.326. В противовес регрессии уклон у бустинга получился в сторону **precision**, микро-усредненное значение которого подросло с 0.46 до 0.67. Но произошло это за счет снижения **recall**, рухнувшего с 0.70 до 0.34. В итоге имеем **F1 score** 0.45, тогда как у регрессии он был 0.55, или на 22% выше.

Особенно стоит отметить обрушение **recall** у небольших жанров, при сохранении схожих значений **precision**. У **реального ТВ** с 0.53 до 0.03, у **ток-шоу** с 0.46 до 0.04, а **фильм-нуар** остался и вовсе нераспознанным моделью. За счет этого имеется снижение макро-усредненного **F1 score** с 0.48 до 0.34, или на 30%.



* + 1. Модель на униграммах и биграммах

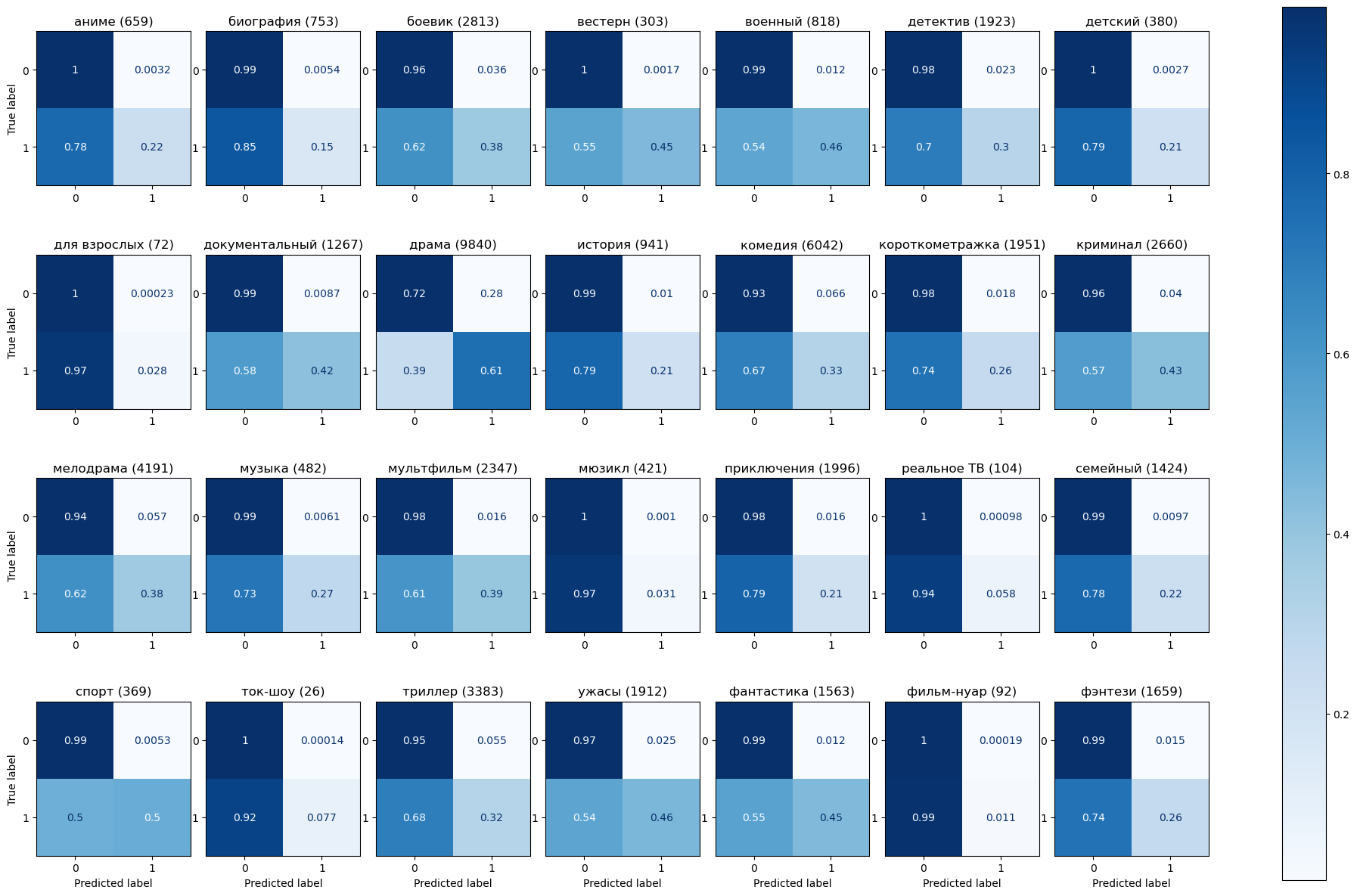
Бустинг на униграммах и биграммах оказался самым затратным по времени из классических подходов. Обучение на CPU заняло 7 часов 27 минут. Тем более досадно, что грандиозного прироста метрик это не принесло.



Не углубляясь в детали, можно отметить, что микро-усредненный **precision** снизился с 0.67 до 0.63, **recall** поднялся с 0.34 до 0.39, потянув за собой вверх **F1 score** с 0.45 до 0.48. Небольшие жанры по-прежнему выглядят неподвластными бустингу, хотя ему и удалось определить один **фильм-нуар**.

Можно заключить, что в случае с бустингом увеличение количества n-грамм не приносит достаточной выгоды, если брать во внимание время обучения и размер итоговой модели, который в моем случае получился 2,35 ГБ, тогда как бустинг на униграммах весит всего 123 МБ. Это естественно сказывается и на скорости инференса.

Перспективным направлением для развития классических подходов мне видится блендинг предсказаний логистической регрессии и бустинга, либо стекинг на двух этих моделях. В таких ансамблях, на мой взгляд, достаточно будет задействовать лишь униграммы.

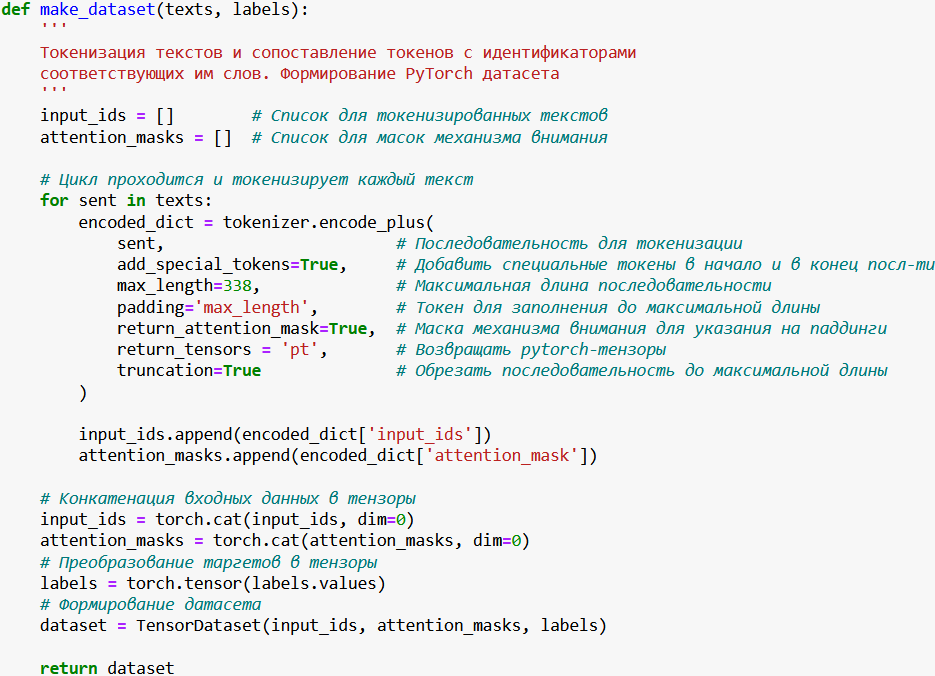


1. Моделирование. Нейросетевые подходы
   1. МЕТОДИКА ОБУЧЕНИЯ

Предобученные трансформенные языковые модели брались из хранилища **Hugging Face**. Для токенизации и дообучения использовались классы **AutoTokenizer** и **AutoModelForSequenceClassification** из библиотеки **transformers**. Из библиотеки **PyTorch** для формирования тензоров и подачи их в модели были задействованы классы **TensorDataset** и **DataLoader**.

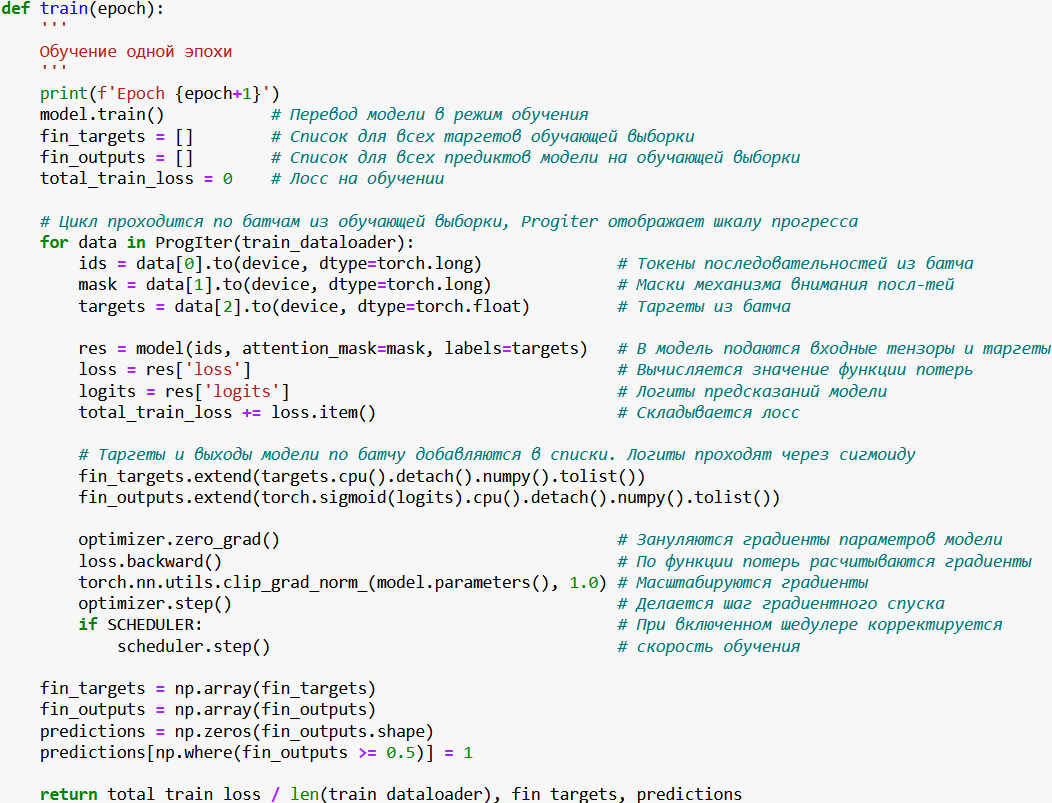
* + 1. Создание датасетов

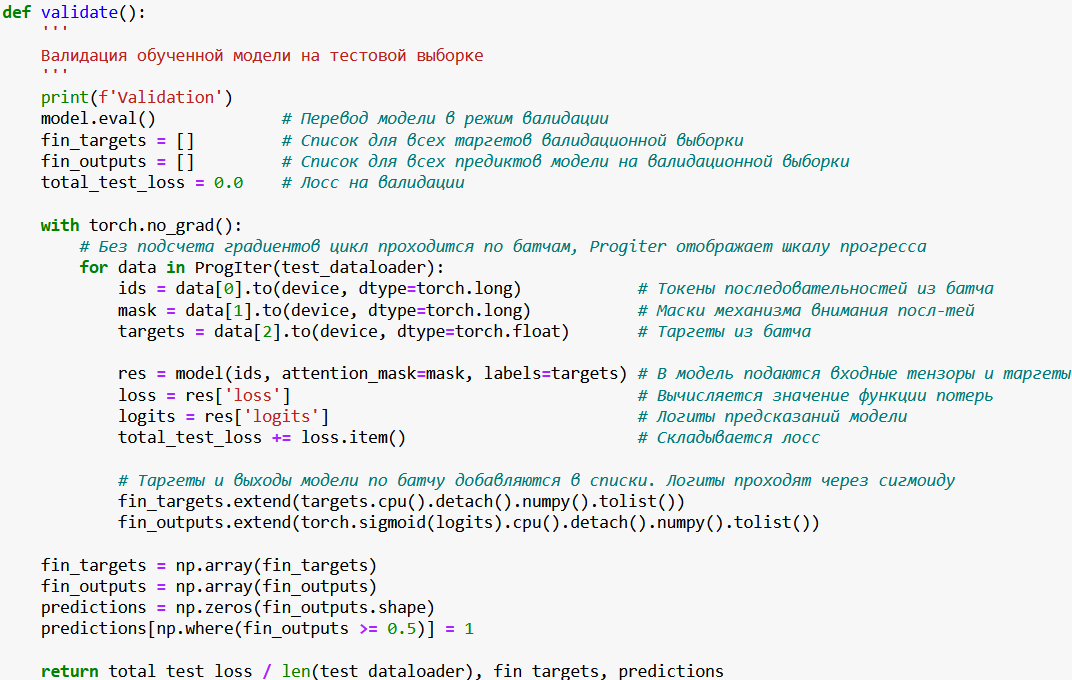
Для создания обучающих и валидационных датасетов была написана функция **make\_dataset**.



* + 1. Функции для обучения и валидации

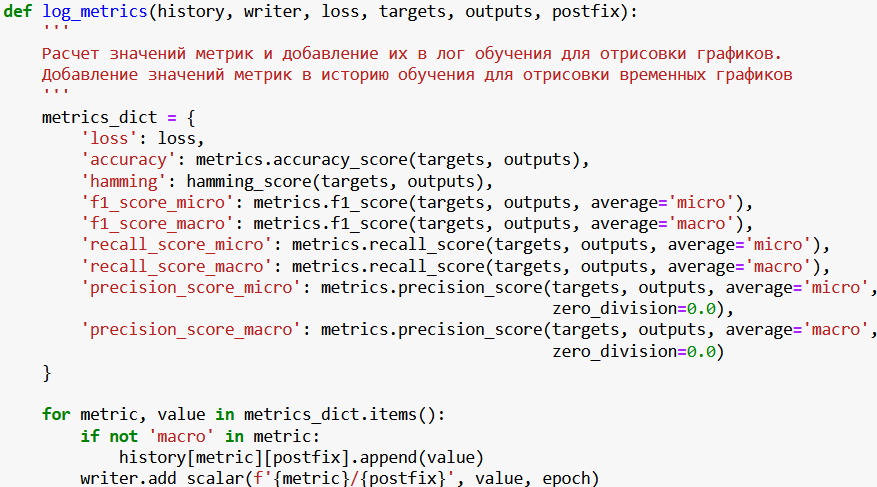
Для обучения моделей была написана функция **train**. Для валидации моделей на тестовой выборке была написана функция **validate**.



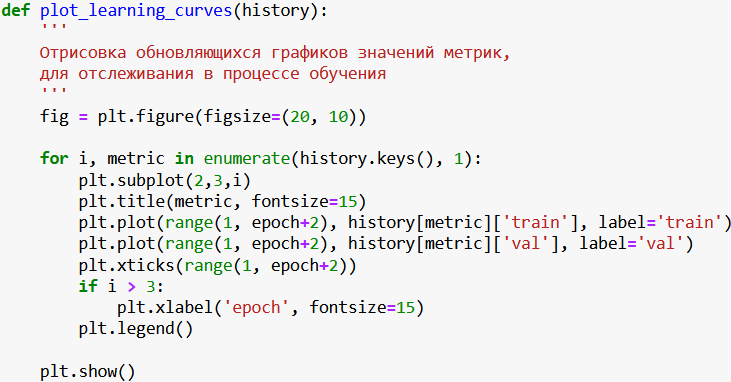


* + 1. Функции логирования и отображения метрик

Для логирования метрик, с целью дальнейшего прочтения их в **Tensorboard** был задействован класс **SummaryWriter** из библиотеки **PyTorch**. Для реализации этого была написана функция **log\_metrics**.

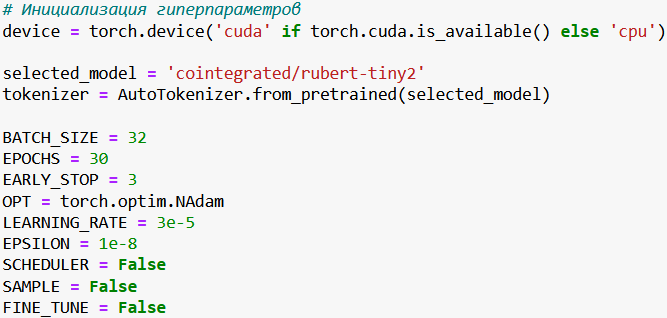


Функция **log\_metrics** также отвечала за наполнение истории для отображения обновляющихся графиков значений метрик в процессе обучения, что осуществлялось с помощью функции **plot\_learning\_curves**.



* + 1. Инициализация и обучение

Функцией потерь для обучения всех моделей послужил класс **BCEWithLogitsLoss** из библиотеки **PyTorch**. Инициализация гиперпараметров, моделей и служебных функций происходила по следующему алгоритму.





В ходе обучения была задействована техника ранней остановки. Метрикой для ранней остановки служил **F1 score**. Почему был сделан выбор в пользу этой метрики, а не значения функции потерь, будет описано в дальнейшем, когда будут продемонстрированы кривые обучения. Обучение моделей происходило по следующему алгоритму.



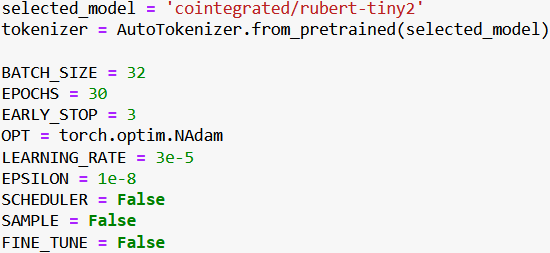
* + 1. Эксперименты

Для подбора гиперпараметров каждой модели эксперименты проводились на 25% данных от обучающей выборки. **Learning rate** выбирался в диапазоне 2\*10-5 – 4\*10-5. Были попытки по использованию шедулера **get\_linear\_schedule\_with\_warmup** из библиотеки **transformers**, в качестве дополнительного механизма регулирования скорости обучения, но это отрицательно сказывалось на качестве работы модели, поэтому решено было от него отказаться. В ходе экспериментов выбирался один из трех оптимизаторов из библиотеки **PyTorch**: **Adam**, **NAdam** или **AdamW**. Размер батча определялся исходя из размеров модели и ожидаемой продолжительности обучения, и изменялся в диапазоне от 8 до 128. Также были попытки обучения не всех параметров моделей, а только классификатора, но обученные таким образом модели оказывались слабее, чем при полноценном дообучении.

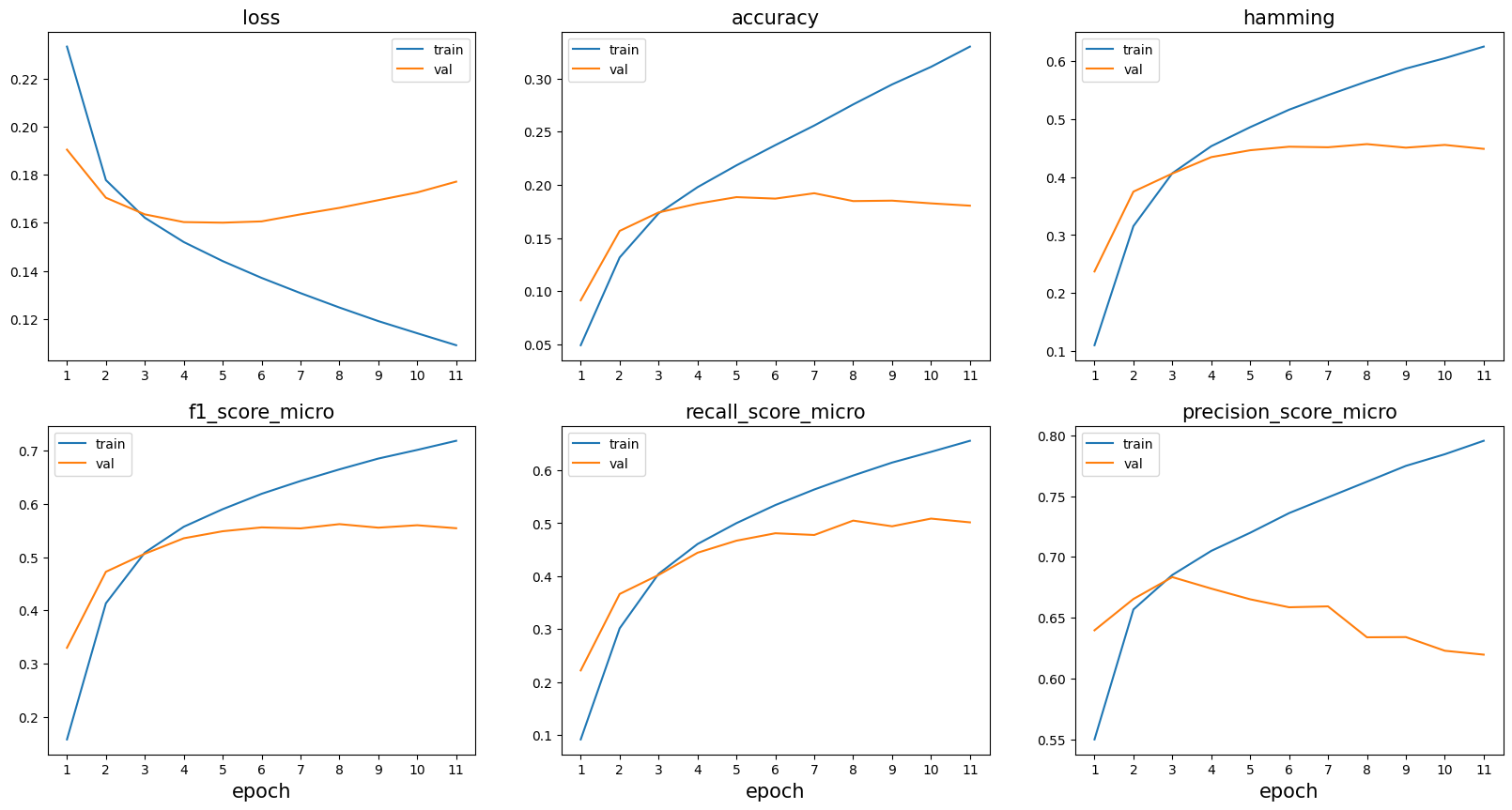
* 1. ТРАНСФОРМЕР RUBERT TINY

Первым трансформером для этого исследования послужил **ruBert tiny2** от **Cointegrated**, обновленная версия небольшого энкодера для русского языка, основанного на модели **BERT**. Это самая небольшая из всех рассматриваемых здесь нейросетевых моделей, у нее 29.4 млн. параметров. Эта и все последующие модели обучались на платформе Kaggle, на предоставляемой в пользование GPU P100.

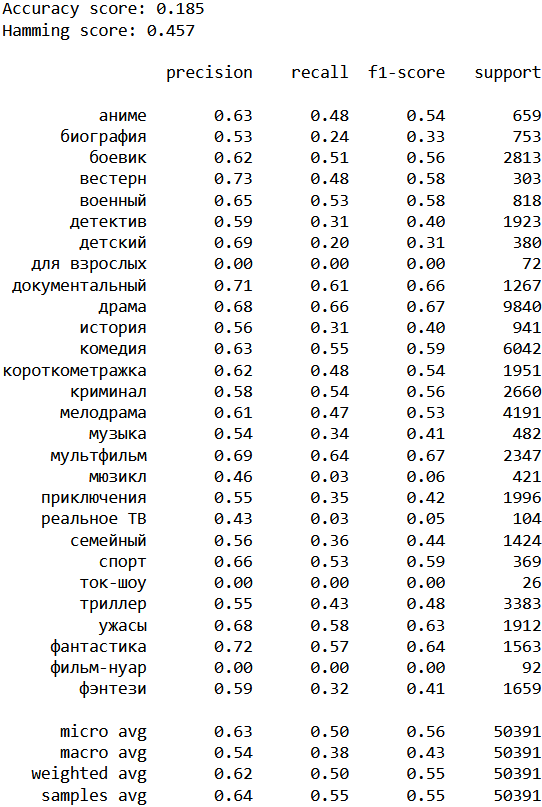
Для обучения использовались следующие гиперпараметры.



Обучение продлилось 11 эпох, при этом лучшей метрики **F1 score**, по которой производилась ранняя остановка, модель достигла уже на 8 эпохе. Если рассматривать приведенные ниже кривые обучения, то заметно, что **loss** на валидационной выборке перестал падать уже после 4 эпохи, хотя продолжался рост **F1 score**, за счет роста **recall**. Рост **recall** сопровождался одновременным снижением **precision**, но за счет того, что **recall** подрастал сильнее, чем падал **precision**, гармоническое среднее между ними продолжало расти вплоть до 8 эпохи. Аналогичное поведение метрик наблюдалось и при обучении других моделей, поэтому было принято решение использовать именно F1 score, в качестве сигнализатора для ранней остановки.

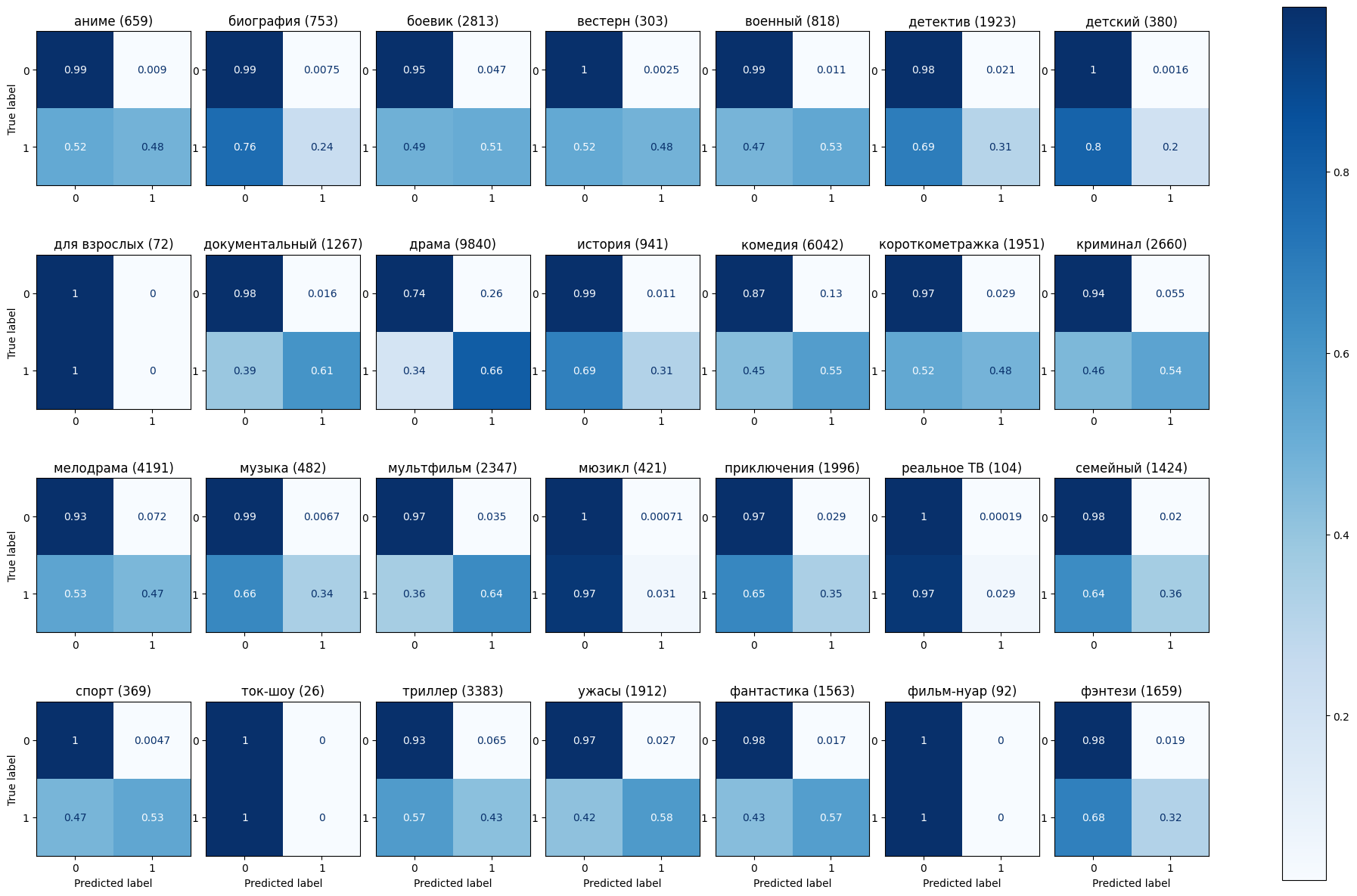


Теперь оценим качество работы обученного **ruBert tiny** на тестовой выборке. В качестве бейзлайна для сравнения я буду использовать логистическую регрессию, обученную на униграммах + биграммах.



Заметен очень серьезный прирост **accuracy score** с 0.103 до 0.185, трансформер, похоже выучил определенные паттерны, которые позволяют ему с большей точностью определять некоторые объекты целиком. При этом **hamming score** у двух моделей вполне сопоставим 0.443 у регрессии и 0.457 у rubert. Уклон у трансформера получился в сторону **precision**, микро-усреднение которого в сравнении с регрессией выросло с 0.50 до 0.63, однако существенное снижение **recall** с 0.68 до 0.50 привело к чуть более низкому итоговому **F1 score**, было 0.58, а стало 0.56. Отмечу, что неоспорим плюсом **ruBert tiny** является его небольшой размер. Модель весит всего 111 МБ, что не много даже в сравнении с регрессией на униграммах + биграммах, вес которой получился 497 МБ.

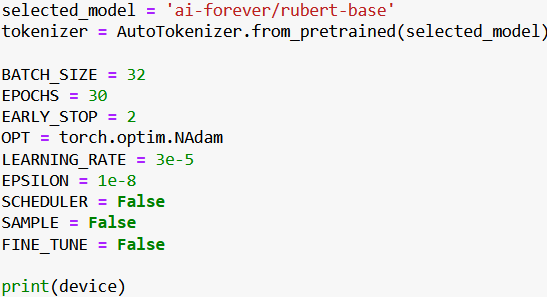
Если обратить внимание на сами классы, то рост **precision** виден практически по всем жанрам, за исключением самых маленьких, представленность которых в тестовом датасете меньше 100 штук – это **для взрослых**, **ток-шоу** и **фильм-нуар**. Нейросеть не смогла правильно идентифицировать ни одного представителя этих классов.



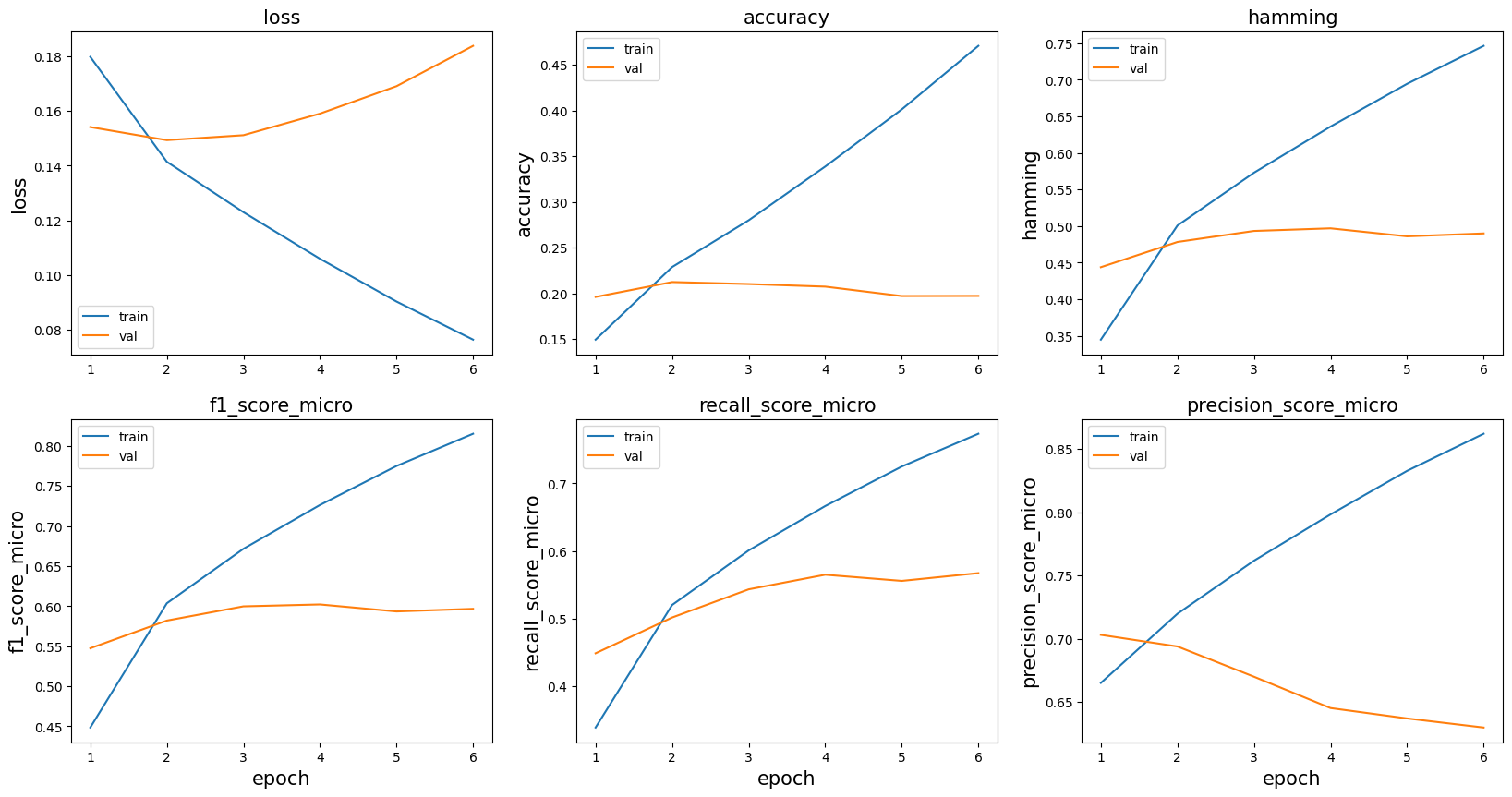
* 1. ТРАНСФОРМЕР RUBERT BASE

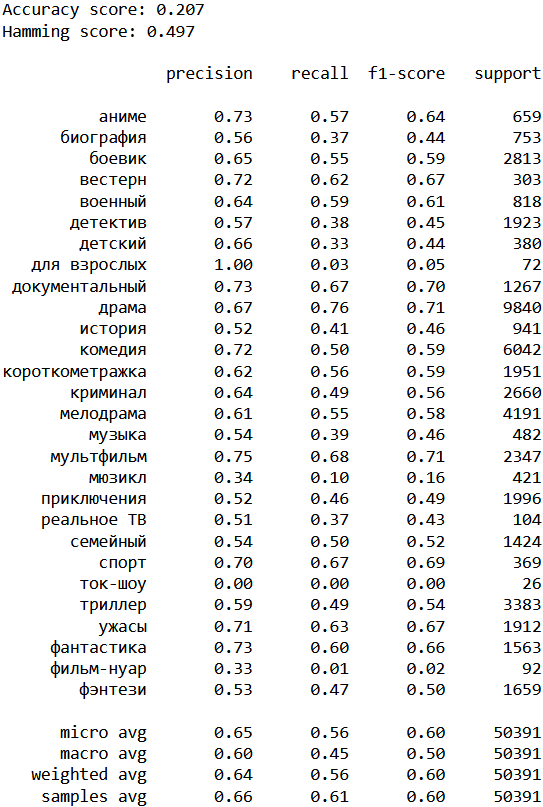
Вторым трансформером будет рассмотрен **ruBert base** от **AI forever**, полновесный энкодер для русского языка. Количество параметров у него уже 178 млн, что в 6 раз больше, чем у **ruBert tiny**.

Для обучения использовались следующие гиперпараметры.



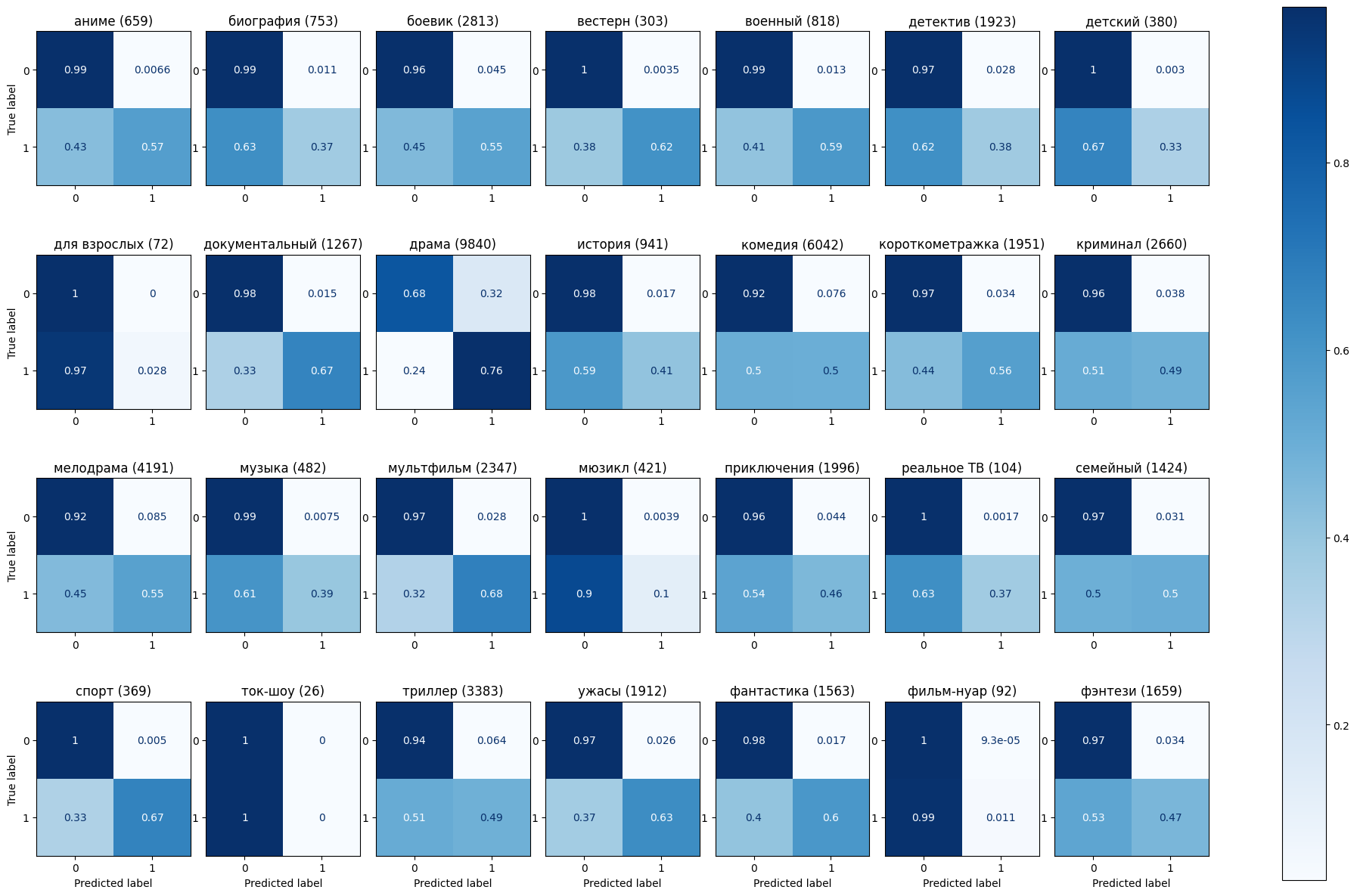
Рост количества параметров естественным образом сказался на продолжительности обучения. Тогда как оптимизация **ruBert tiny** на 11 эпохах отняла 46 минут, обучение **ruBert base** на 6 эпохах стоило уже 5 часов и 11 минут. По графикам видно, что **loss** на валидационной выборке достиг минимума уже на 2 эпохе, хотя **F1 score** еще продолжал улучшаться и был максимальным на 4 эпохе.





Оценив модель на тестовой выборке видно, что относительно **ruBert tiny** ощутимо подросли все метрики. Accuracy score поднялся с 0.185 до 0.207, а **hamming score** с 0.457 до 0.497 – по значению этой метрики уже можно заявить, что модель верно оценивает жанры в половине случаев, что выглядит впечатляюще, учитывая, что всего классов 28. Улучшились и микро-усредненные **precision** до 0.65, и **recall** до 0.56, и **F1 score** до 0.60.

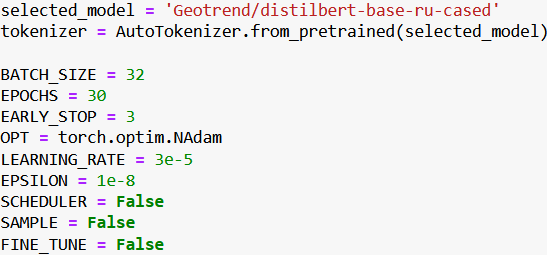
Тем не менее ситуация с небольшими жанрами по-прежнему выглядит неудовлетворительной. Есть незначительное улучшение у **фильм-нуар** и **для взрослых**, а **ток-шоу** остался для модели незамеченным. Здесь возникает соображение о том, что для настолько мало представленных жанров с целью их обогащения разумно было бы прибегнуть к аугментациям.



* 1. ТРАНСФОРМЕР distilbert base ru

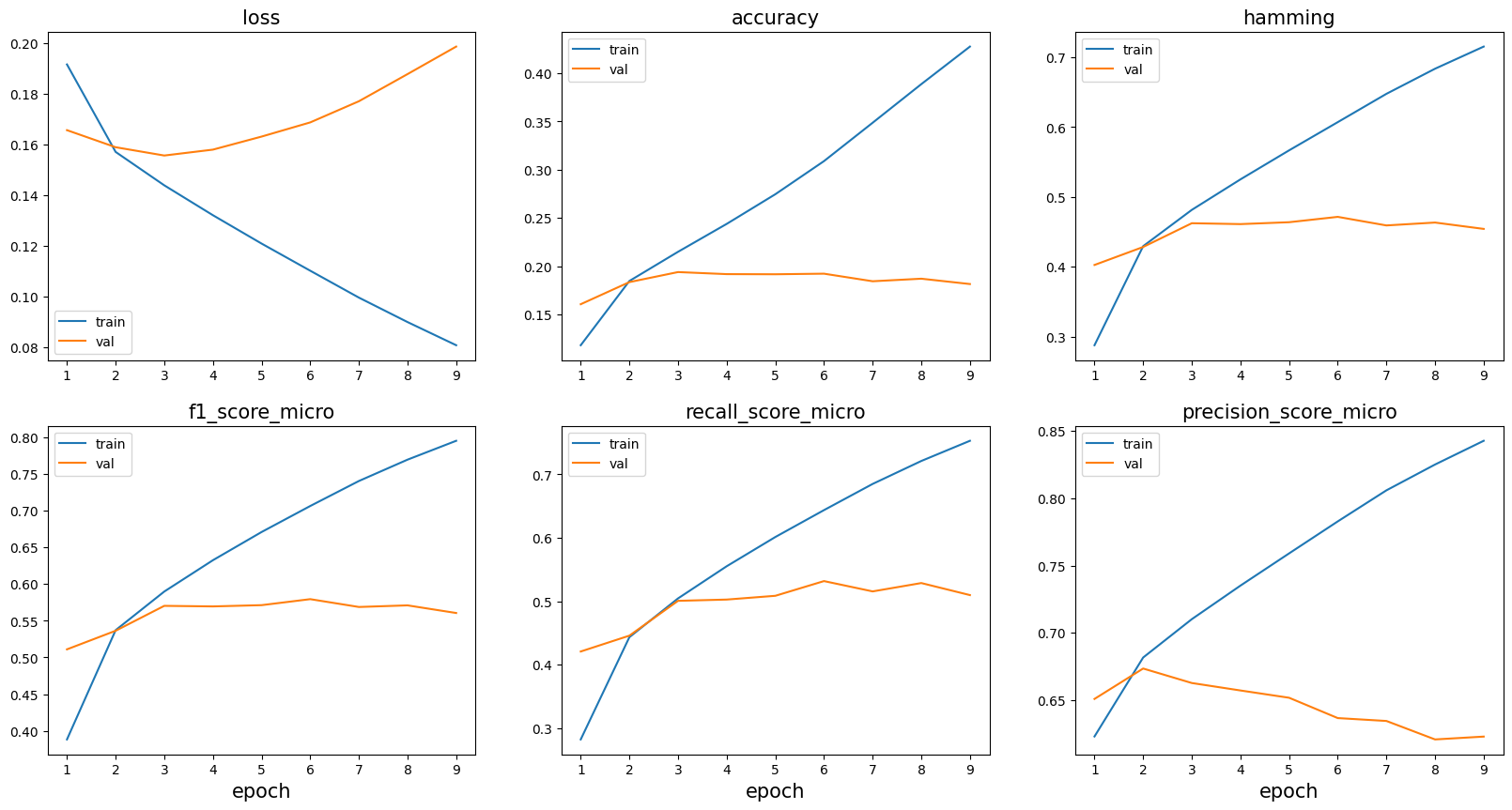
Третьим обученным трансформером стал **DistilBERT base ru cased** от **Geotrend**, версия мультиязычного энкодера для русского языка c количеством параметров 54.5 млн, что более чем в 3 раза меньше, чем у предыдущей модели **ruBert base**.

Для обучения использовались следующие гиперпараметры.

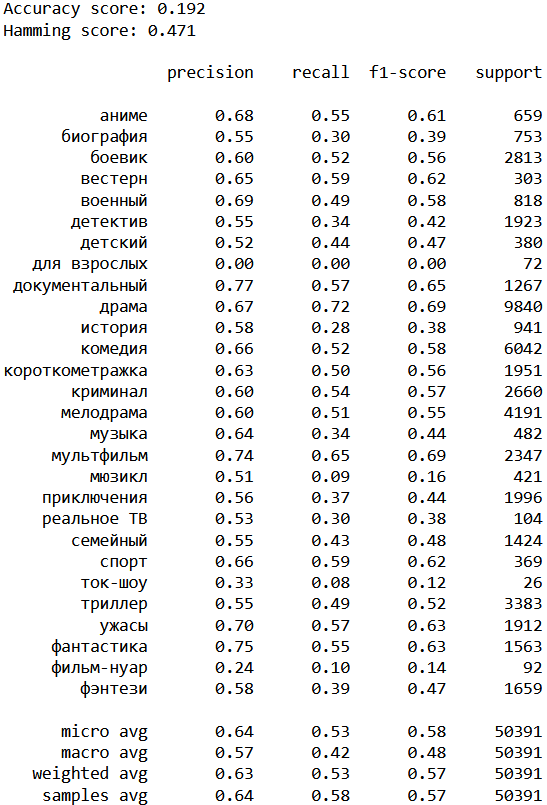


**DistilBERT** создавался в качестве облегченного варианта **BERT**, с прицелом на оптимизацию обучения за счет уменьшения размера и увеличения скорости, при попытке сохранить производительность[[6]](#footnote-6). Это заметно и по процессу обучения, 9 эпох оптимизации **DistilBERT** заняло 3 часа 55 минут, в сравнении с 6 эпохами за более чем 5 часов у **ruBert base**. На соответствующих графиках опять же видно, что **F1 score** еще некоторое время продолжал улучшаться вплоть до 6 эпохи, после достижения минимума значения функции потерь уже на 3 эпохе.

Здесь я хочу отметить еще один интересный момент, а именно, что кривые **hamming score** на графиках обучения моделей имеют абсолютно такую же форму, что и кривые **F1 score**, при том, что сами значения метрик различаются. То есть для ранней остановки в качестве сигнализатора можно было бы использовать и **hamming score**.



Теперь оценим качество работы обученного **DistilBERT** на тестовой выборке.



По всем своим метрикам **DistilBERT** занимает промежуточное положение между **ruBert tiny** и **ruBert base**. В качестве интересной особенности можно отметить, что **DistilBERT** немного лучше справился с микро-жанрами **ток-шоу** (F1=0.12) и **фильм-нуар** (F1=0.14). У **ruBert base** было **ток-шоу** (F1=0.00) и **фильм-нуар** (F1=0.02). Конечно, это все равно не очень много, и заметно хуже, чем было у логистической регрессии.

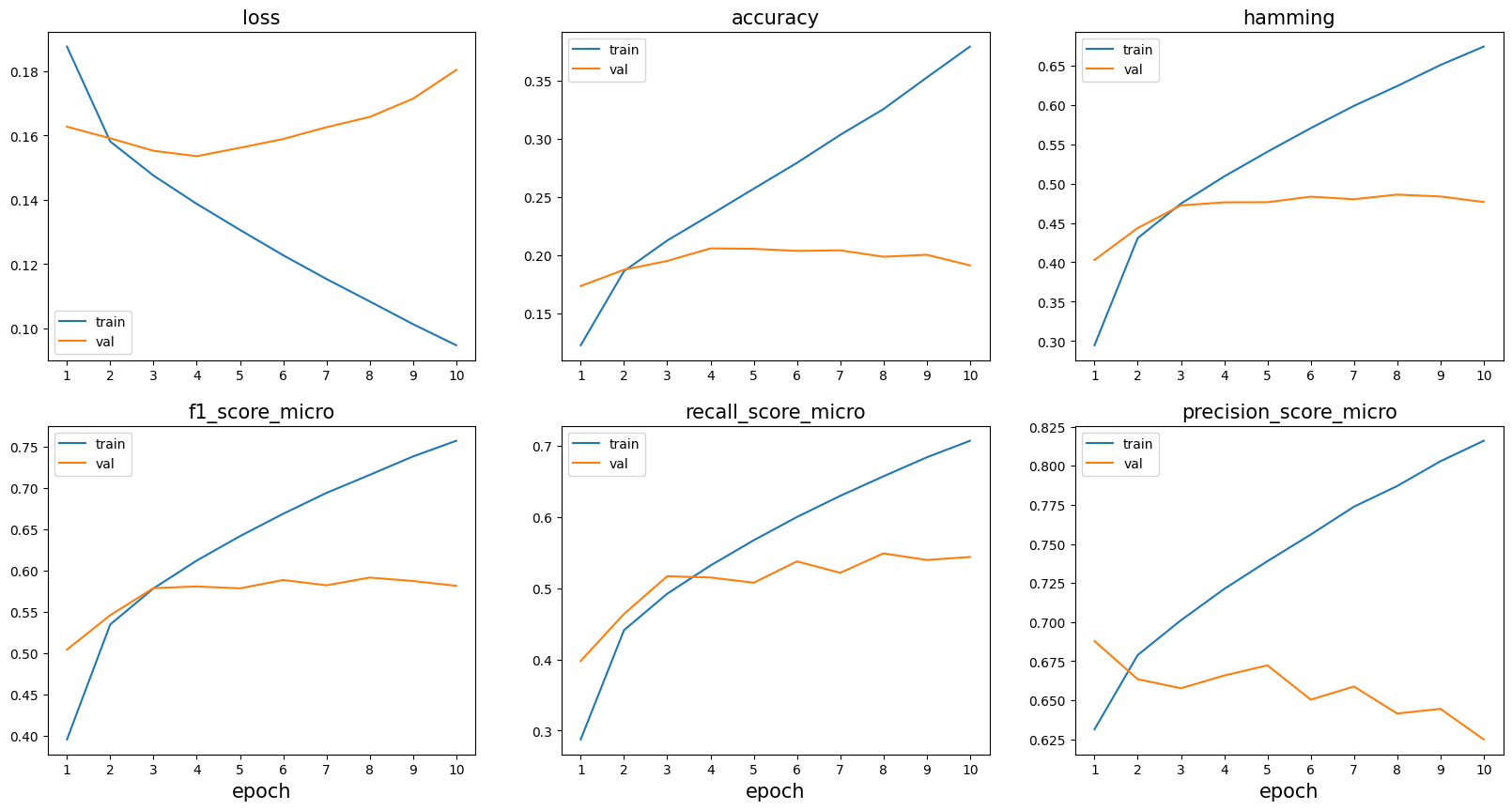


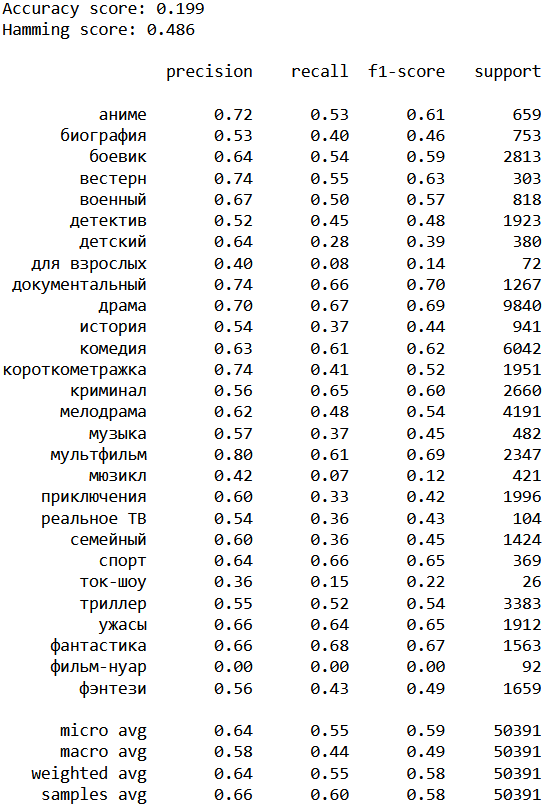
* 1. ТРАНСФОРМЕР XLM-ROBERTA BASE

Последним трансформером, обученным в рамках этого исследования, стала модель **XLM-RoBERTa base** от **FacebookAI** – это единственная рассматриваемая здесь мультиязычная модель. Говоря о параметрах, **RoBERTa** становится по этому показателю лидером исследования, их количество 279 млн. Для обучения использовались следующие гиперпараметры.



Обучение **RoBERTa** продлилось 10 эпох, что при таком внушительном размере модели заняло 10 часов 31 минуту. Своего максимума метрика **F1 score** достигла на 8 эпохе, и это представляется длинной дистанцией, если вспомнить, что схождение **ruBert base** было выполнено за 4 эпохи.





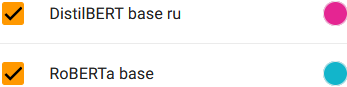
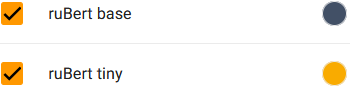
Увеличенное вдвое, в сравнении с **ruBert base** время, затраченное на обучение **RoBERTa** не обернулось приростом метрик. Хотя модель по своим показателям и приблизилась к лидеру, ни в одной из оцениваемых метрик превзойти его не удалось. Единственным успехом относительно других трансформеров является чуть большее внимание к двум микро-жанрам: **для взрослых** (F1=0.14) и **ток-шоу** (F1=0.22).

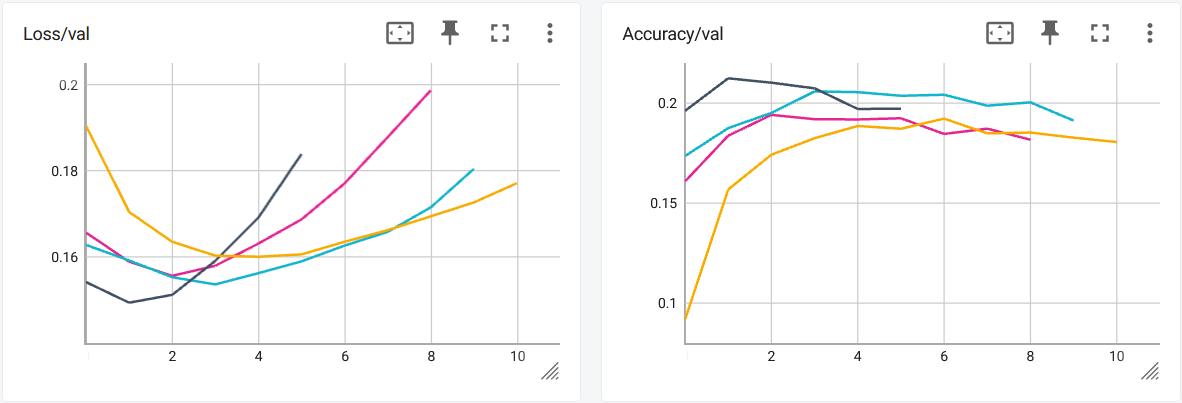
Я предполагаю, что такие относительно скромные показатели **RoBERTa** в моем исследовании могут быть связаны с тем, что модель мультиязычная, а также с тем, что могла потребоваться более тонкая настройка гиперпараметров. Второе естественным образом осложнялось ввиду большого размера архитектуры и следовательно большего количество времени на проведение экспериментов.

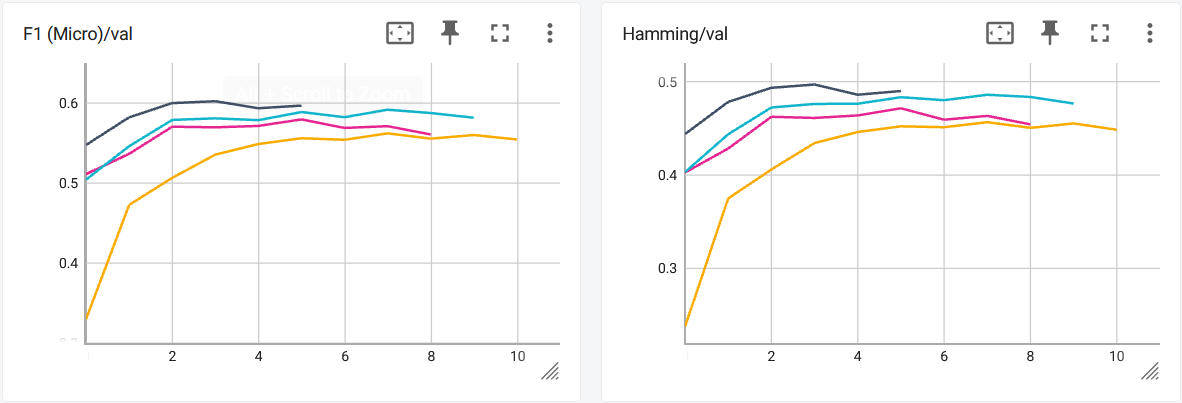


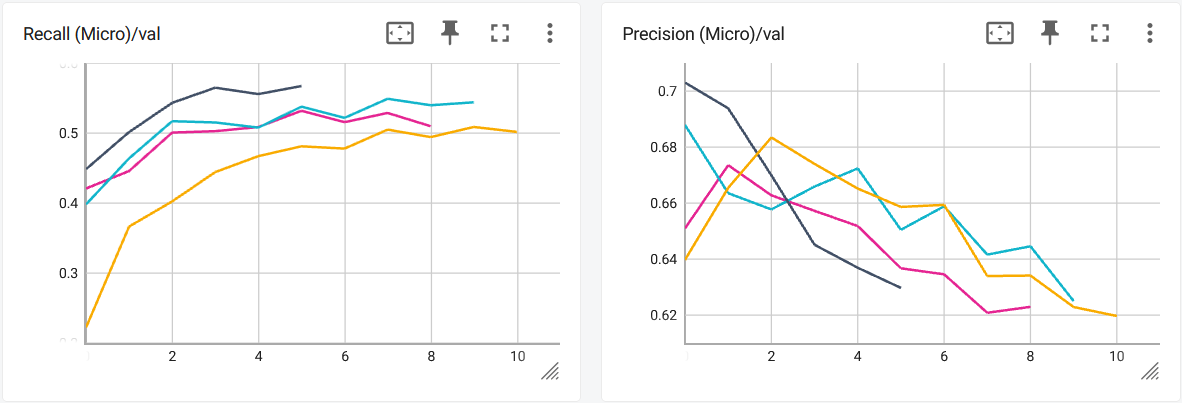
1. Сравнение моделей
   1. СРАВНЕНИЕ КРИВЫХ ОБУЧЕНИЯ ТРАНСФОРМЕННЫХ МОДЕЛЕЙ

Сравнение качества работы трансформенных моделей и некоторых специфик процесса их обучения я буду производить на основании графиков метрик, полученных на валидационной выборке. Показатели для удобства сравнения были загружены в **Tensorboard**.







По всем ключевым метрикам, кроме **precision** заметно преимущество **ruBert base** над остальными моделями. В целом можно сказать, что в плане качества выстроилась иерархия, в которой за **ruBert base** идет **RoBerta base**, затем с небольшим отставанием **DistilBERT**, и, наконец, **ruBert tiny**. Хотя **ruBert tiny** по всем метрикам стартует с заметно более низкой базы, в ходе обучения эта модель постепенно приближается к показателям остальных архитектур. Некоторые кривые (**F1 score**, **hamming score**, **recall**) у моделей **RoBerta base** и **DistilBERT** расположены очень близко друг к другу, что свидетельствует в пользу эффективности **DistilBERT**, количество параметров у которого более чем в 5 раз меньше, чем у **RoBerta base**.

Еще раз отмечу тот факт, что минимизация **loss** достигается раньше, чем максимизация метрик **F1 score**, **hamming score** и **recall**, и это хорошо видно по графикам. Хотя минимальный **loss** и соответствует максимальному **accuracy**, эта метрика на мой взгляд не является ключевой для решения поставленной задачи multi-label классификации. Важнее мне представляются **F1 score** и **hamming score**, максимизация которых происходит за счет более скорого роста **recall** на фоне периодического снижения **precision** – это также общее поведение для всех моделей. И еще на общих графиках отчетливо видна близость по смыслу метрик **F1 score** и **hamming score**, формы их кривых очень сильно схожи у всех моделей, а разница значений выглядит постоянной, на уровне 0.10.

* 1. ОБЩЕЕ СРАВНЕНИЕ ВСЕХ ПОДХОДОВ

Для сравнения между собой всех моделей, обученных в рамках этого исследования как в классических, так и в нейросетевых подходах, а также для формулирования итоговых выводов привожу сводную таблицу основных характеристик и метрик. Значения **F1 score**, **Recall** и **Precision** здесь приведены в микро-усреднении.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model name** | **Accuracy score** | **Hamming score** | **F1 score** | **Recall** | **Precision** | **Loss** | **Epochs** | **Train time, min.** | **Inference time, sec.** | **Params, M** | **Size, MB** |
| LogReg 1-gram | 0.073 | 0.415 | 0.554 | **0.699** | 0.459 |  |  | 2 | **0.73** |  | **24** |
| LogReg 1+2-gram | 0.103 | 0.443 | 0.577 | 0.683 | 0.499 |  |  | 21 | 1.69 |  | 497 |
| CatBoost 1-gram | 0.130 | 0.326 | 0.449 | 0.338 | **0.666** |  |  | 41 | 5.53 |  | 123 |
| CatBoost 1+2-gram | 0.130 | 0.353 | 0.482 | 0.390 | 0.630 |  |  | 447 | 115 |  | 2406 |
| ruBert tiny | 0.185 | 0.457 | 0.562 | 0.505 | 0.634 | 0.166 | 8 | 46 | 17 | 29.4 | 111 |
| ruBert base | **0.207** | **0.497** | **0.602** | 0.565 | 0.645 | **0.159** | 4 | 311 | 238 | 178 | 680 |
| DistilBERT base | 0.192 | 0.471 | 0.579 | 0.532 | 0.637 | 0.169 | 6 | 235 | 120 | 54.5 | 207 |
| RoBERTa base | 0.199 | 0.486 | 0.592 | 0.549 | 0.642 | 0.166 | 8 | 631 | 259 | 279 | 1055 |

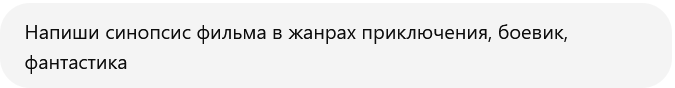
Очевидно, что максимальный **accuracy score** достигается с применением нейросетей, хотя **hamming score**, выдаваемый логистической регрессией выглядит вполне конкурентоспособным. Лидером рейтинга как по **hamming score**, так и по **F1 score** является **ruBert base**. В плане сбалансированности качества можно признать эту модель наилучшей несмотря на то, что максимальным **recall** обладает логистическая регрессия на униграммах, а максимальным **precision** CatBoost на униграммах.

Чтобы говорить о функциональности различных моделей необходимо обратить внимание на их вес и время инференса, которое соответствует здесь времени, затраченному на классификацию всего тестового датасета (21536 фильмов). TFIDF и логистическая регрессия выглядят не только хорошим бейзлайном в плане метрик, но также являются и самым быстродействующим решением, которое к тому же занимает минимум места – всего 24 МБ для модели на униграммах. При этом, для такого подхода вероятно будет достаточно трудно существенно превысить базовые показатели метрик. Мне не кажется, что оптимизация гиперпараметров даст ощутимый качественный прирост. Здесь пространство для улучшений мне видится в ансамблировании, а именно блендинг предсказаний логистической регрессии и бустинга, либо стекинг на двух этих моделях.

Гораздо большее пространство для экспериментов открывается в нейроетевых подходах. Классификаторы, построенные на эмбеддингах, производимых трансформенными энкодерами способны показывать превосходное качество при должной настройке. Даже небольшой **ruBert tiny** уже демонстрирует очевидный прирост ключевых метрик. Но здесь необходимо помнить о том, что количество параметров, а соответственно вес модели и время, требуемое на работу нейросети пока что растут экспоненциально относительно прироста в качественных показателях. Поэтому для промышленного использования таких высококачественных подходов необходимо либо обладать достаточными инфраструктурными ресурсами, либо учитывать ограничения в их быстродействии.

* 1. СРАВНЕНИЕ НА ПРИМЕРАХ

В качестве еще одного испытания для оцениваемых моделей я решил протестировать их на данных, порожденных другой моделью. Я попросил GPT-4o написать синопсисы несуществующих фильмов в заданных мною жанровых рамках, например:



Не пытаясь качественно оценить каждый из алгоритмов, приведу здесь результаты их работы. Сгенерированные синопсисы приведены в сокращенном варианте (1 абзац из 5).

**"Стражи Галактики: Пробуждение"**

**Жанры**: боевик, приключения, фантастика

В далёком будущем, когда человечество распространилось по всей галактике, межзвездная империя оказывается на грани краха. Могущественный инопланетный артефакт, известный как "Пульс Звёзд", внезапно активируется, нарушая баланс сил и пробуждая древних и могущественных существ, спящих в глубинах космоса…

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **LogReg 1-gram** | **LogReg 1+2-gram** | **CatBoost 1-gram** | **CatBoost 1+2-gram** | **ruBert tiny** | **ruBert base** | **DistilBERT base** | **RoBERTa base** |
| аниме, **боевик**, мультфильм, **приключения**, семейный, **фантастика**, фэнтези | аниме, **боевик**, мультфильм, **приключения**, семейный, **фантастика**, фэнтези | **боевик**, мультфильм, **приключения**, **фантастика** | **боевик**, мультфильм, **приключения**, **фантастика**, фэнтези | **боевик**, мультфильм, **приключения**, **фантастика** | **боевик**, **приключения**, **фантастика** | **боевик**, **приключения**, **фантастика** | **боевик**, **приключения**, **фантастика** |

1. <https://wiki.cloudfactory.com/docs/mp-wiki/metrics/hamming-score> [Hamming score] [↑](#footnote-ref-1)
2. <https://www.kinopoisk.ru/lists/movies/?page=1> [Раздел сайта с каталогом фильмов] [↑](#footnote-ref-2)
3. [https://habr.com/ru/companies/otus/articles/808521](https://habr.com/ru/companies/otus/articles/808521/) [Рабочий парсер для Кинопоиска] [↑](#footnote-ref-3)
4. [https://habr.com/ru/articles/503420](https://habr.com/ru/articles/503420/) [Лемматизируй это быстрее (PyMorphy2, PyMystem3 и немного магии)] [↑](#footnote-ref-4)
5. <https://github.com/trent-b/iterative-stratification?tab=readme-ov-file> [iterative-stratification] [↑](#footnote-ref-5)
6. <https://habr.com/ru/articles/680986> [Все, что нужно знать об ALBERT, RoBERTa и DistilBERT] [↑](#footnote-ref-6)