

# Пояснительная записка

# к дипломному проекту

# на тему:

# Предсказание тематических категорий новостей по их содержанию

Автор: Трофимов Павел

Группа: DSU-60

Ментор: Яблонева Тора

Презентация: [Дипломная работа.Презентация](https://docs.google.com/presentation/d/1VGzD3roFJqspsK0UnXWlqhX7CuykYW1e65I7m6W82uA/edit?usp=sharing)

Репозиторий: [Дипломная работа. Github](https://github.com/TrofimovPaul/NetologyThesis)

Датасеты: [Исходный датасет (800 000 строк)](https://www.kaggle.com/datasets/yutkin/corpus-of-russian-news-articles-from-lenta/data), [Обработанный датасет (70889 строк)](https://www.kaggle.com/datasets/trofimovpaul/news-datasets-train-test)

2025

# Содержание

[Введение 4](#_Toc201840860)

[Актуальность проблемы 4](#_Toc201840861)

[Предложение по решению проблемы и анализ подхода 4](#_Toc201840862)

[Достоинства подхода 4](#_Toc201840863)

[Основные недостатки подхода 5](#_Toc201840864)

[Постановка задачи 6](#_Toc201840865)

[Описание данных и их особенностей 7](#_Toc201840866)

[Исходный датасет. Общая информация 7](#_Toc201840867)

[Основные характеристики датасета 7](#_Toc201840868)

[Анализ и подготовка датасета 7](#_Toc201840869)

[Данные исходного датасета 9](#_Toc201840870)

[Анализ данных 9](#_Toc201840871)

[Обработка данных 13](#_Toc201840872)

[Датасет для исследования. Общая информация 13](#_Toc201840873)

[Метрики 15](#_Toc201840874)

[Правильность (Accuracy) 15](#_Toc201840875)

[Точность (Precision) 15](#_Toc201840876)

[Полнота (Recall) 15](#_Toc201840877)

[F1-мера (F1-measure) 15](#_Toc201840878)

[Матрица ошибок (Confusion matrix) 16](#_Toc201840879)

[Похожесть (Identity) 17](#_Toc201840880)

[Расстояние Хэмминга (Hamming distance) 18](#_Toc201840881)

[Лемматизация текстов 20](#_Toc201840882)

[Результаты лемматизации 20](#_Toc201840883)

[Разделение на обучающий и тестовый датасет 22](#_Toc201840884)

[Описание обучения 23](#_Toc201840885)

[Классические методы машинного обучения 23](#_Toc201840886)

[LogisticRegression с TfidfVectorizer на униграммах 24](#_Toc201840887)

[LogisticRegression с TfidfVectorizer на униграммах и биграммах 26](#_Toc201840888)

[CatBoostClassifier с TfidfVectorizer на униграммах 28](#_Toc201840889)

[Нейросетевые методы машинного обучения 30](#_Toc201840890)

[Создание датасетов 30](#_Toc201840891)

[Функции для обучения и валидации 31](#_Toc201840892)

[Функции логирования и отображения метрик 32](#_Toc201840893)

[Инициализация 33](#_Toc201840894)

[Обучение модели 34](#_Toc201840895)

[Испытания 34](#_Toc201840896)

[Модели 35](#_Toc201840897)

[Трансформер ruBERT tiny2 (Cointegrated) 35](#_Toc201840898)

[Трансформер ruBERT base (AI forever) 37](#_Toc201840899)

[Трансформер DistilBERT (Geotrend) 40](#_Toc201840900)

[Трансформер XLM-RoBERTa (FacebookAI) 42](#_Toc201840901)

[Сравнение моделей 46](#_Toc201840902)

[Сравнение нейросетевых моделей 46](#_Toc201840903)

[ruBERT base 46](#_Toc201840904)

[XLM-RoBERTa 46](#_Toc201840905)

[DistilBERT 46](#_Toc201840906)

[ruBERT tiny2 46](#_Toc201840907)

[Сравнение классических и нейросетевых моделей 47](#_Toc201840908)

[Примеры 48](#_Toc201840909)

[Список литературы 51](#_Toc201840910)

# Введение

## Актуальность проблемы

В современном информационном пространстве объём новостных источников и статей растёт с каждым днём. Помимо официальных новостных источников в сети появляется всё больше людей, которые ведут собственные каналы в соцсетях и мессенджерах, делятся новостями, аналитикой и личными мнениями.

В таких условиях, когда информационные потоки становятся все более децентрализованными, а авторы не уделяют достаточно внимания структурированию своих публикаций, пользователю становится все сложнее быстро находить действительно важную и интересующую его информацию.

Обычный поиск по ключевым словам часто приводит к неудовлетворительным результатам, среди которых легко потеряться. По этой причине требуется решение для классификации новостных статей.

## Предложение по решению проблемы и анализ подхода

Перспективным решением для эффективной навигации видится поиск новостей по тегам, проставленным на основании содержания.

### Достоинства подхода

Основными достоинствами тегов являются:

* Возможность группировать публикации вне зависимости от источника (крупные новостные платформы, пользовательские медиа) по темам, событиям или ключевым персонам
* Обеспечение доступа к полному списку материалов в рамках интересующего вопроса
* Упрощенный поиск и фильтрация контента
* Снижение возможности упустить важные детали
* Повышение роста вовлеченности аудитории
* Поддержка актуальности новостного потока, обусловленная быстрой ориентацией даже в самых насыщенных информационных лентах

### Основные недостатки подхода

Основными недостатками тегов являются:

* Сложность назначения тегов для составителя, ввиду при написании публикаций и необходимости продумывания ключевых тем, описываемых в публикациях
* Сложность разметки / назначения тегов сторонним человеком, ввиду необходимости погружения в контекст, постоянных концентрации и внимания на темах, описываемых в публикациях
* Высокие трудозатраты на назначение тегов при больших объёмах и количестве публикаций
* Пересечение тегов для публикаций из разных тематических областей

# Постановка задачи

Исходя из описанных выше недостатков, можно заключить, что ручное назначение тегов является не самым продуктивным процессом. В то же время, описанные выше достоинства показывают неоспоримый положительный эффект от использования тегов.

С учётом этих двух выводов, появляется необходимость найти решение по автоматическому назначению тегов для новостных публикаций. Данную задачу можно успешно решить с использованием современных методов машинного обучения (классических и нейросетевых). Данными для обучения, в качестве основы, должны выступать новостные публикации, включающие содержание и тему, назначенные ранее составителями.

Таким образом, основная задача и цель могут быть сформулированы следующим образом:

**Задача:**

1. Подготовить данные по новостным публикациям для дальнейшего обучения моделей.
2. Разработать модели машинного обучения для решения задачи многоклассовой классификации, в частности автоматического назначения тегов для новостных публикаций на основе их содержания
3. Провести оценку и сравнение результатов по обученным моделям

**Цель работы:**

Целью работы является разработка и выбор наилучшего решения по автоматическому назначению корректных тегов для новостных публикаций для упрощения поиска и фильтрации публикаций по интересующим пользователя вопросам/темам.

# Описание данных и их особенностей

## Исходный датасет. Общая информация

В качестве исходных данных был взят датасет [**News dataset from Lenta.Ru**](https://www.kaggle.com/datasets/yutkin/corpus-of-russian-news-articles-from-lenta/data) с новостными публикациями на русском языке.

### Основные характеристики датасета

* **Представленный период публикаций:**
* Сентябрь 1999 года - Декабрь 2019 года
* **Количество уникальных записей:**
* Более 800 000
* **Размер:**
* Более 2 Гб
* **Поля и описание:**
  + **url** (Ссылка на публикацию, уникальные значения)
  + **title** (Заголовок публикации)
  + **text** (Содержание публикации)
  + **topic** (Тема публикации)
  + **tags** (Теги (подтемы) публикации)
  + **date** (Дата публикации)

### Анализ и подготовка датасета

При первичном анализе датасета было выяснено следующее:

1. Данные, представленные в полях **“url”**, **“title”**, не требуются для дальнейшего исследования.
2. В поле **“date”** основное значение имеет год публикации.
3. Для корректной идентификации строк требуется ввести дополнительный столбец **“id”**.
4. Поля **“topic”** и **“tags”** несут в себе информацию практически об одном и том же, на основании чего было принято решение оставить только поле **“tags”** с предварительной обработкой:
   * Для строк, у которых отсутствовали данные по полю **“tags”**, в поле **“tags”** дублировались данные из поля **“topic”**.
   * Строки, у которых отсутствовали данные по полям **“topic”** и **“tags”**, были исключены из дальнейшего исследования.

Таким образом, конечный список полей исходного датасета выглядит следующим образом:

* **id** (Идентификатор публикации)
* **text** (Содержание публикации)
* **tags** (Теги публикации)
* **year** (Год публикации)

После очистки датасета от строк с пустыми полями **“topic”** и **“tags”** количество записей сократилось примерно до **739 000**.

Количество уникальных тегов: 104

## Данные исходного датасета

### Анализ данных

На основании данных, представленных в исходном датасете, можно вывести некоторую статистику. Интересной видится информация о количестве публикаций по различным тематическим категориям (тегам), представленная на диаграмме ниже см. рис. 1)

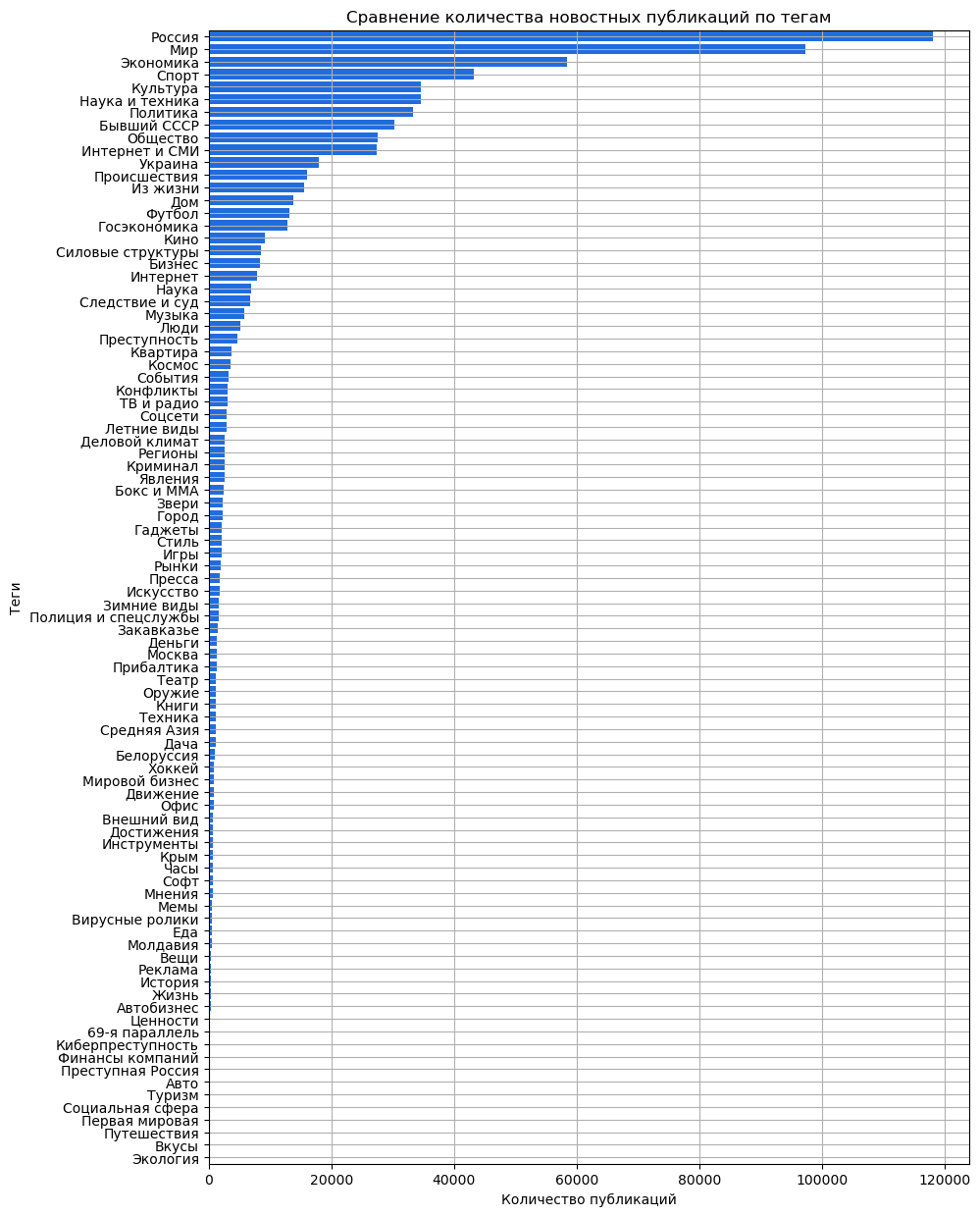


Рис. 1. Сравнение количества новостных публикаций по тегам

**При анализе диаграммы можно сделать следующие выводы:**

* График демонстрирует широкий спектр освещаемых тем - от геополитики до таких специализированных областей как кино, музыка и спорт. Это указывает на стремление к тематической диверсификации контента новостного источника. Другими словами, это указывает на то, что данный новостной источник является общеновостным, не затрагивающим какую-то конкретную узкую тему.
* Распределение на графике показывает весьма классическую модель медиа-потребления: небольшое количество категорий генерирует основную долю контента, в то время как множество нишевых тем получают ограниченное освещение. Это соответствует принципу Парето (20/80) в медиа-индустрии. Притом некоторые нишевые темы являются дочерними по отношению к другим более популярным. (например, Театр - Культура)
* На графике четко прослеживаются приоритеты в освещении новостей. Политическая и общественная тематика (Россия, Мир) доминирует над специализированными направлениями. Это типично для массовых медиа, которые фокусируются на темах широкого общественного интереса

**При рассмотрении отдельных категорий можно обратить внимание на то, что:**

* Публикации с категорией “Россия” занимают вершину списка, значительно опережая все остальные категории, что, очевидно, обусловлено отношением датасета к новостному источнику из России.
* Высокий медийный интерес в анализируемый период также наблюдается для таких категорий, как “Мир”, “Экономика”, “Спорт” и “Культура”. Такое распределение можно объяснить тем, что людей, помимо новостей об их родном месте, чаще всего интересуют новости из других стран, денежная ситуация и культурно-массовые мероприятия.
* В среднем сегменте по количеству публикаций находятся такие категории как:
  + Наука и техника
  + Здоровье
  + Бывший СССР
  + Общество
  + Интернет и СМИ
* Такие категории можно назвать бытовыми. Техника, как и медицина, в последние годы развивается, общество часто интересует его история и события, связанные с ним самим.
* Нижняя часть графика показывает специализированные и узконаправленные темы с относительно небольшим количеством публикаций, включая:
  + Искусство
  + Зимние виды спорта
  + Полиция и спецслужбы
  + Заповедники
  + Прибалтика

Также интересно рассмотреть количество публикаций по годам для каждой тематической категории (см. рис. 2). Следует обратить внимание, что в легенде графика представлены не все подписи категорий, но это, в целом, никак не мешает анализу.

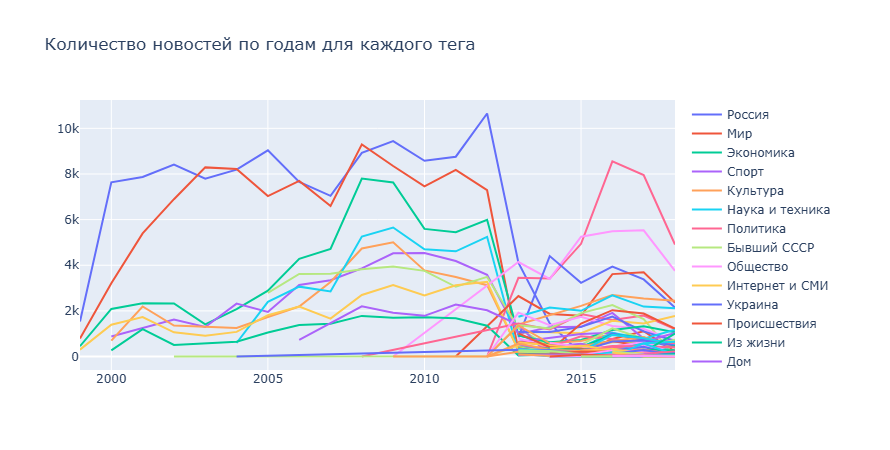


Рис. 2. Сравнение количества публикаций по годам для каждой тематической категории (тега)

По графику хорошо прослеживается скудное разделение новостных публикаций по категориям в период с 1999 года по 2012 год, что говорит об обобщении нескольких категорий в одну и нецелесообразности выделения каких-то конкретных малопопулярных категорий. Такое достаточно часто наблюдалось на тот момент времени у большинства новостных источников. Также можно предположить, что ориентация новостного источника на освещение обширного количества тем на тот момент времени не предполагалась.

В 2012-2013 годах появляется огромное количество новых категорий, основные категории “Россия”, “Мир” и другие ранее популярные утрачивают свои позиции, вероятно, ввиду декомпозиции на более узкие категории. Такое поведение, возможно, связано с активным внедрением системы хэштегов во все имеющиеся сервисы и социальные сети.

На графике также можно отметить:

* Сезонность для категории “Спорт”, что, несомненно, связано с проведением Олимпийских игр каждые два года
* Всплеск новостей с категорией “Мир” в 2003-2004 годах, связанный с какими-то крупными событиями
* Увеличение количества публикаций с тегами “Россия”, “Мир”, “Экономика” в 2008-2009 году, что, очевидно, связано с мировым кризисом, наступившим примерно в этот период
* Увеличение количества публикаций с тегами “Политика”, “Общество”, “Украина” в 2014-2016 годах, связанное с активным освещением событий, разворачивающихся вокруг Украины

# Обработка данных

## Датасет для исследования. Общая информация

Так как размер исходного датасета и количество уникальных записей достаточно велики, было принято решение создать датасет на основании исходного с меньшим количеством записей для ускорения обработки и обучения моделей.

Для этого были выполнены следующие действия:

1. Создание списка категорий (категория “69-я параллель” исключена из исследования) с количеством записей больше 50. Была использована формула get\_tags\_count:

def get\_tags\_count(engine, num=50):  
 # получение списка тегов, где количество записей больше 50  
   
 if num > 0:  
 query = f'''  
 select tags, count(id) from news\_table nt   
 group by tags  
 having count(id) > {int(num)}  
 order by count(id) desc  
 '''  
 df\_tags = pd.read\_sql\_query(query, con = engine)  
 return df\_tags

**Количество уникальных тегов после фильтрации: 89**

1. Выбор 1000 уникальных строк по каждому тегу с сортировкой по полю “id” по возрастанию.

# проход по полученному списку тегов и выбор первых 1000 строк по каждому тегу при сортировке по id  
sum\_tags = 0  
for tag in df\_tags.tags:  
 query = f'''  
 select \* from news\_table nt   
 where tags = '{tag}'  
 order by id  
 limit 1000  
 '''  
 # сохранение данных по каждому тегу в датафрейм  
 df\_tag\_for\_save = pd.read\_sql\_query(query, con = engine)  
 len\_df = len(df\_tag\_for\_save)  
 sum\_tags += len\_df  
 # сохранение полученного датафрейма по каждому тегу в csv файл с указанием количества строк  
 df\_tag\_for\_save.to\_csv(f'D:\\netology\_diplom\\final\\csv\_files\\data\_for\_analyze\_by\_tags\\{tag}\_{len\_df}.csv', index =   
 False)  
  
# получение списка файлов из папки data\_for\_analyze\_by\_tags  
csv\_list = os.listdir('D:\\netology\_diplom\\final\\csv\_files\\data\_for\_analyze\_by\_tags')  
f = 0  
for file in csv\_list:  
 print(f"\r{file[:-4]}:", end="")  
 if f == 0:  
 df\_file = pd.read\_csv('D:\\netology\_diplom\\final\\csv\_files\\data\_for\_analyze\_by\_tags\\' + file)  
 df\_full\_data = df\_file.drop(df\_file.index, inplace = True)  
 else:  
 df\_file = pd.read\_csv('D:\\netology\_diplom\\final\\csv\_files\\data\_for\_analyze\_by\_tags\\' + file)  
 df\_full\_data = pd.concat([df\_full\_data, df\_file])  
 print(f"\r{file[:-4]}:", 'OK', end="\n")  
 f += 1  
df\_full\_data.to\_csv('D:\\netology\_diplom\\final\\csv\_files\\full\_data.csv', index = False)  
  
df\_full\_data = pd.read\_csv('D:\\netology\_diplom\\final\\csv\_files\\full\_data.csv')  
df\_full\_data['year'] = pd.to\_datetime(df\_full\_data['date']).dt.year  
df\_fd = df\_full\_data.drop(columns = ['url','title','topic','date'])  
df\_fd.drop\_duplicates()  
  
df\_text\_tags = df\_fd.drop(columns = ['id', 'year'])  
  
df\_text\_tags\_final = pd.get\_dummies(df\_text\_tags, prefix = ['tags'], columns=['tags'])  
for i in df\_text\_tags\_final.columns[1:]:  
 df\_text\_tags\_final[i] = df\_text\_tags\_final[i].astype(int)  
  
list\_col = df\_text\_tags\_final.columns[:1].to\_list()  
for i in df\_text\_tags\_final.columns[1:].to\_list():  
 list\_col.append(i[5:])  
list\_col  
df\_text\_tags\_final.columns = list\_col  
  
df\_text\_tags\_final.to\_csv('D:\\netology\_diplom\\final\\csv\_files\\news\_data\_lemma.csv', index = False)  
df\_text\_tags\_final  
#df\_text\_tags\_final=pd.read\_csv('D:\\netology\_diplom\\final\\csv\_files\\news\_data\_lemma.csv')

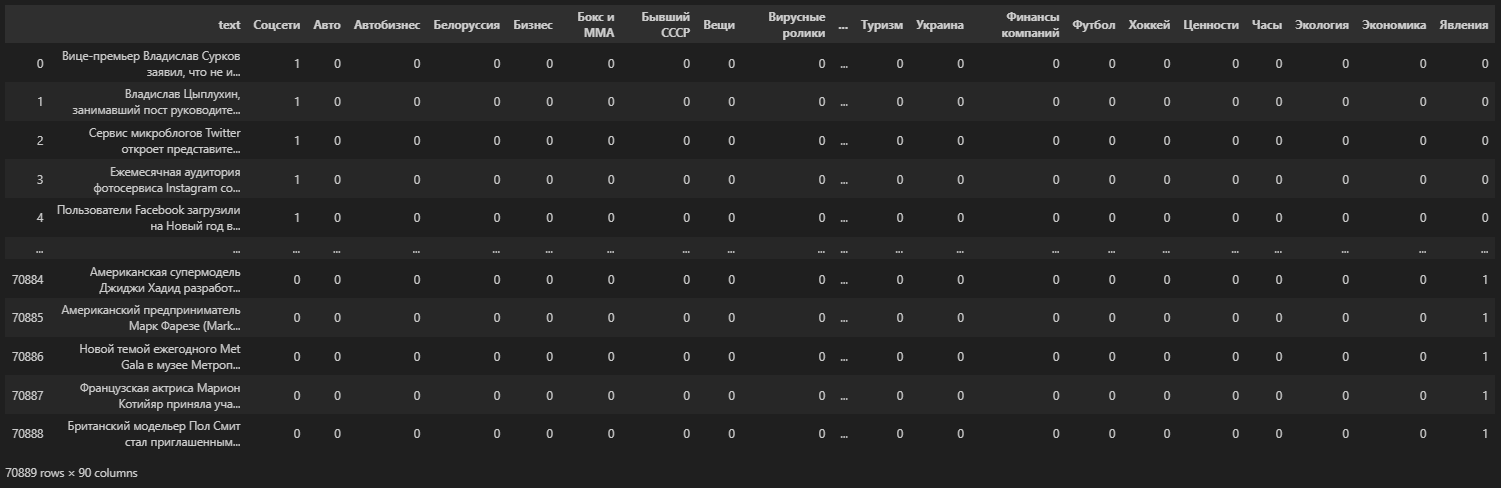


Рис. 3. Датасет для исследования

**Общее количество строк после обработки: 70889**

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 70889 entries, 0 to 70888  
Data columns (total 7 columns):  
# Column Non-Null Count Dtype   
--- ------ -------------- -----   
0 id 70889 non-null int64   
1 url 70889 non-null object  
2 title 70889 non-null object  
3 text 70889 non-null object  
4 topic 70889 non-null object  
5 tags 70889 non-null object  
6 date 70889 non-null object  
dtypes: int64(1), object(6)  
memory usage: 3.8+ MB

## Метрики

### Правильность (Accuracy)

Доля объектов, для которых правильно предсказан класс. Объект считается классифицированным верно, если предсказанный вектор полностью совпадает с таргетом.

Стоит иметь в виду, что Accuracy имеет несколько недостатков: - Не учитывает дисбаланс классов - Не учитывает цену ошибки на объектах разных классов (ошибочно положительное определение класса не так критично, как ошибочно отрицательное).

### Точность (Precision)

Точность показывает долю правильно предсказанных положительных объектов среди всех объектов, предсказанных положительным классом. Иначе говоря, в рамках поставленной задачи, точность по каждому тегу показывает, сколько из определённых нами объектов с таким тегом действительно относятся к этому тегу.

Общее значение точности равно среднему арифметическому значению точности по всем тегам.

### Полнота (Recall)

Полнота показывает долю правильно найденных положительных объектов среди всех объектов положительного класса. Иначе говоря, в рамках поставленной задачи, полнота по каждому тегу показывает, какую долю объектов с таким тегом удалось выявить.

Общее значение полноты равно среднему арифметическому значению полноты по всем тегам.

### F1-мера (F1-measure)

F1-мера представляет среднее гармоническое точности и полноты. F1-мера предполагает одинаковую важность Precision и Recall.

Общее значение F1-меры равно среднему арифметическому значению F1-меры по всем тегам.

### Матрица ошибок (Confusion matrix)

Матрица, состоящая из комбинаций, которые могут получаться при сопоставлении ответов алгоритма/модели и истинных меток объекта:

* TP — истинно-положительные объекты (TruePositive) — объект представляет собой класс 1 и алгоритм его идентифицирует как класс 1
* FP — ложно-положительные объекты (FalsePositive) — объект представляет собой класс0, алгоритм его идентифицирует как класс 1(незначительная ошибка)
* TN — истинно-отрицательные объекты (TrueNegative) — объект представляет собой класс0 и алгоритм его идентифицирует как класс 0
* FN — ложно-отрицательные объекты (FalseNegative) — объект представляет собой класс 1, алгоритм его идентифицирует как класс 0. (грубая ошибка)

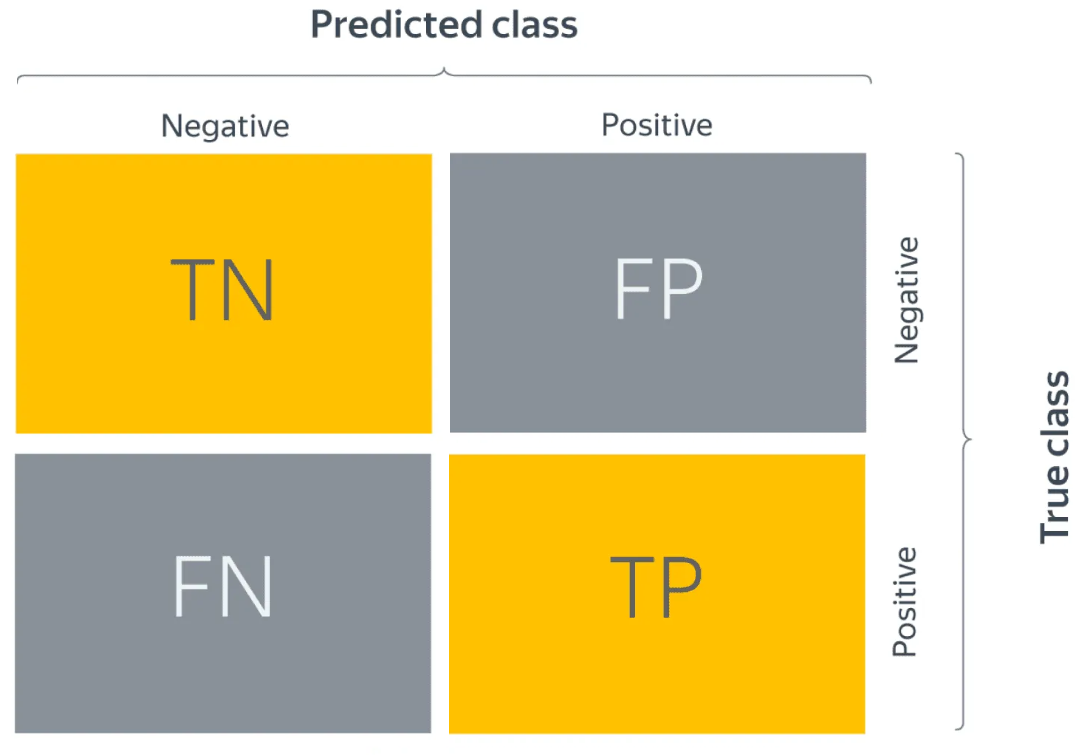


Рис. 4. Схема матрицы несоответствия

def cm\_show(y\_test, y\_pred, nrows=10, ncols=5, figsize\_w=30, figsize\_h=60, wspace=0.2, hspace=0.4):  
 '''  
 Вывод матрицы несоответствия для каждой тематической категории в формате:   
 [['TN', 'FP'],  
 ['FN', 'TP']]  
 y\_test - тестовые целевые данные  
 y\_pred - предсказанные целевые данные  
 nrows - количество строк в сетке матриц несоответствия (для корректного отображения сетки должно  
 работать уравнение: nrows = figsize\_h / 6)  
 ncols - количество столбцов в сетке матриц несоответствия (для корректного отображения сетки должно  
 работать уравнение: ncols = figsize\_w / 6)  
 figsize\_w - ширина полотна для вывода сетки матриц несоответствия (для корректного отображения сетки должно  
 работать уравнение: figsize\_h = 6 \* ncols)  
 figsize\_h - высота полотна для вывода сетки матриц несоответствия (для корректного отображения сетки должно  
 работать уравнение: figsize\_h = 6 \* nrows)  
 wspace - расстояние по ширине между отдельными ячейками сетки матриц несоответствия  
 hspace - расстояние по высоте между отдельными ячейками сетки матриц несоответствия  
 '''  
   
 start = time.time()  
 y\_true = y\_test.to\_numpy()  
 axis = plt.subplots(nrows, ncols, figsize=(figsize\_w, figsize\_h))[1]  
 axis = axis.ravel()  
  
 # многоклассовая матрица несоответствий  
 mlcm = multilabel\_confusion\_matrix(y\_true, y\_pred)  
   
 for tag\_num in range(len(y\_test.columns)):  
 # Если надо отобразить проценты.   
 # Так как в рамках задачи наибольший интерес вызывает информация о точном предсказании (y\_true = 1, y\_pred = 1)  
 # или о грубой ошибке при предсказании (y\_true = 1, y\_pred = 0) категории, имеет смысл выводить информацию для  
 # дальнейшего анализа в виде доли верных и неверных предсказаний для целевого значения = 1.   
 mlcm\_percent = mlcm[tag\_num] / mlcm[tag\_num].sum(axis=1).reshape(2, -1)  
   
 true0\_row = mlcm[tag\_num].sum(axis=1)[0] # количество TN, FP предсказаний (нецелевое значение = 0)  
 true1\_row = mlcm[tag\_num].sum(axis=1)[1] # количество TP, FN предсказаний (целевое значение = 1)  
   
 mlcm\_percent\_display = ConfusionMatrixDisplay(mlcm\_percent, display\_labels=[0, 1])   
 mlcm\_percent\_display.plot(ax=axis[tag\_num],cmap=plt.cm.GnBu, values\_format="0.3f")  
 mlcm\_percent\_display.ax\_.set\_title(f'{y\_test.columns[tag\_num]} (TL0: {true0\_row}, TL1:{true1\_row})')   
  
 plt.subplots\_adjust(wspace=wspace, hspace=hspace)  
 plt.show()  
 return print(f'Время выполнения: {round(time.time() - start,3)} с')

### Похожесть (Identity)

Расстояние Хэмминга в своём стандартном виде показывает количество позиций, в которых соответствующие символы двух объектов одинаковой длины различаются. В рамках поставленной задачи был рассмотрен вывод доли похожести объектов при помощи расстояния Хэмминга.

Стоит учитывать, что Identity имеет несколько недостатков:

* При увеличении количества классов приближается к единице, что усложняет интерпретацию метрики
* Не даёт понимания, в чём конкретно похожи объекты. Например, для пар значений “10001” / “11111” и “10001” / “10110” похожесть будет одинаково равна 0,4, но в первой паре количество единиц равно 5, а во второй паре значений равно 3.

def identity(y\_test, y\_pred):  
 '''  
 Расчёт похожести объектов через расстояние Хэмминга  
 y\_test - тестовые целевые данные  
 y\_pred - предсказанные целевые данные  
 Вывод среднего значения похожести объектов по всем элементам   
 '''  
 y\_true = y\_test.to\_numpy()  
  
 if (len(y\_true) != len(y\_pred)):  
  
 raise Exception('Объекты должны быть одинаковой длины')  
 identity\_list = []  
 for elem in range(len(y\_true)):  
 if (len(y\_true[elem]) != len(y\_pred[elem])):  
  
 raise Exception('Элементы должны быть одинаковой длины')  
 # Инициализация переменной расстояния Хэмминга  
 dist\_counter = 0  
 for n in range(len(y\_true[elem])):  
 # Изменение расстояния Хэмминга при наличии разницы между объектами  
 if y\_true[elem][n] != y\_pred[elem][n]:  
   
 dist\_counter += 1  
   
 len\_elem = len(y\_true[elem])  
 # Расчёт доли похожести элементов двух объектов через Расстояние Хэмминга(dist\_counter/len\_elem - доля непохожести двух объектов)  
 identity = round(1 - dist\_counter/len\_elem, 3)  
 identity\_list.append(identity)  
  
 return round(np.mean(identity\_list),5)

### Расстояние Хэмминга (Hamming distance)

По причине того, что стандартный расчёт расстояния Хэмминга не несёт большого смысла для рассматриваемой задачи и Identity обладает рядом перечисленных недостатков, было принято решение доработать формулу расчёта расстояния Хэмминга для более точной интерпретации и понимания различий между объектами.

def hamming\_distance(y\_test, y\_pred):  
 '''  
 Расчёт доработанного расстояния Хэмминга с ориентацией на целевое значение = 1  
 y\_test - тестовые целевые данные  
 y\_pred - предсказанные целевые данные  
 Вывод среднего значения расстояния Хэмминга по всем элементам   
 '''  
 y\_true = y\_test.to\_numpy()  
  
 if (len(y\_true) != len(y\_pred)):  
 raise Exception('Объекты должны быть одинаковой длины')  
 distance\_list = []  
   
 for elem in range(len(y\_true)):  
 if (len(y\_true[elem]) != len(y\_pred[elem])):  
 raise Exception('Элементы должны быть одинаковой длины')  
  
 indices\_1\_y\_true = set(np.where(y\_true[elem])[0])  
 indices\_1\_y\_pred = set(np.where(y\_pred[elem])[0])  
   
 if len(indices\_1\_y\_true) == 0 and len(indices\_1\_y\_pred) == 0:  
 dist\_counter = 1  
  
 else:  
 dist\_counter = len(indices\_1\_y\_true.intersection(indices\_1\_y\_pred)) / float(len(indices\_1\_y\_true.union(indices\_1\_y\_pred)))  
  
 distance\_list.append(dist\_counter)  
 return round(np.mean(distance\_list),5)

Для отображения метрик используется модуль metrics библиотеки sklearn (методы **accuracy\_score**, **confusion\_matrix**, **classification\_report**) и описанные выше функции **hamming\_distance**, **identity**.

# Лемматизация текстов

Лемматизация текстов реализована с использованием библиотек PyMystem3, NLTK. Проблемы со скоростью работы библиотеки PyMystem3 были решены группировкой текстов по 1000 записей через разделитель и дальнейшей разгруппировкой. Время, затраченное на лемматизацию: ~ 9,5 минут.

## Результаты лемматизации

**ИСХОДНЫЙ ТЕКСТ:** Бывший госсекретарь США Хиллари Клинтон завела аккаунт в Twitter. Первый твит Клинтон опубликовала вечером 10 июня по московскому времени. В своей первой и пока единственной записи она поблагодарила создателей блога Texts From Hillary (SMS от Хиллари) «за вдохновение». На момент написания заметки на микроблог бывшего госсекретаря США подписалось более 320 тысяч пользователей. Среди пяти аккаунтов, на которые подписана сама Хиллари Клинтон — ее дочь Челси и муж Билл. 42 президент США зарегистрировался в Twitter в апреле 2013 года. Его появлению в сервисе микроблогов также предшествовала шутка. В начале апреля ведущий юмористической передачи «The Colbert Report» Стивен Кольбер (Stephen Colbert) завел пародийный аккаунт @prezbillyjeff, которым предложил воспользоваться самому Биллу Клинтону. Блог Texts From Hillary, который упомянула Клинтон, был создан интернет-деятелями Адамом Смитом (Adam Smith) и Стейси Ламбе (Stacy Lambe). В нем размещались комиксы, в которых обыгрывалась фотография экс-госсекретаря, сидящей в самолете и читающей текст с экрана телефона. Создатели блога представляли, с кем и о чем могла переписываться Клинтон. Проект был запущен в апреле 2012 года и просуществовал всего неделю. Тем не менее, Хиллари Клинтон обратила на него внимание. Героиня комиксов встретилась со Смитом и Ламбе и поблагодарила их за шутки.

**ЛЕММАТИЗИРОВАННЫЙ ТЕКСТ:** бывший госсекретарь сша хиллари клинтон заводить аккаунт twitter первый твит клинтон опубликовывать вечер июнь московский время свой первый пока единственный запись поблагодарить создатель блог texts from hillary sms хиллари вдохновение момент написание заметка микроблог бывший госсекретарь сша подписываться тысяча пользователь среди пять аккаунт который подписывать хиллари клинтон дочь челси муж билл президент сша зарегистрироваться twitter апрель год появление сервис микроблог также предшествовать шутка начало апрель ведущий юмористический передача the colbert report стивен кольбер stephen colbert заводить пародийный аккаунт prezbillyjeff который предлагать воспользоваться билл клинтон блог texts from hillary который упоминать клинтон создавать интернет деятель адам смит adam smith стейси ламбе stacy lambe немой размещаться комикс который обыгрываться фотография экс госсекретарь сидеть самолет читать текст экран телефон создатель блог представлять мочь переписываться клинтон проект запускать апрель год просуществовать неделя менее хиллари клинтон обращать внимание героиня комикс встречаться смит ламбе поблагодарить шутка

Можно отметить, что из лемматизированного текста исключены числа, специальные символы (например: скобки, тире), предлоги и т.п.

# Разделение на обучающий и тестовый датасет

При разделении датасета на обучающий и тестовый был использован класс MultilabelStratifiedShuffleSplit из библиотеки iterative-stratification, который обеспечивает пропорциональное представление каждой категории в обучающем и тестовом датасете. В обучающую выборку перенесены 56711 публикаций, что составляет 80% данных, в тестовую выборку перенесены 14178 публикаций, соответствующие 20% данных.

splitter = MultilabelStratifiedShuffleSplit(test\_size=0.2, random\_state=42)  
train\_index, test\_index = next(splitter.split(X=df\_text\_tags\_final, y=df\_text\_tags\_final[list\_col[2:]]))  
df\_text\_tags\_final.iloc[train\_index][['text\_lemma']+list\_col[2:]].to\_csv('news\_train.csv', index=False)  
df\_text\_tags\_final.iloc[test\_index][['text\_lemma']+list\_col[2:]].to\_csv('news\_test.csv', index=False)

# Описание обучения

При обучении моделей использовались классические и нейросетевые методы машинного обучения. Матрицы несоответствий для каждой категории по каждой модели представлены в приложениях.

## Классические методы машинного обучения

В качестве основных классических моделей были выбраны следующие модели:

1. LogisticRegression с векторизатором TfidfVectorizer на униграммах из библиотеки sklearn. Также использовался MultiOutputClassifier

pipe = Pipeline([('tfidf', TfidfVectorizer(analyzer='word',ngram\_range=(1, 1))),  
 ('logreg', MultiOutputClassifier(estimator=LogisticRegression(max\_iter=10000,class\_weight='balanced', multi\_class='multinomial',))),])  
pipe.fit(X\_train, y\_train)  
  
Время обучения: 2min 32s

1. LogisticRegression с векторизатором TfidfVectorizer на униграммах и биграммах из библиотеки sklearn. Также использовался MultiOutputClassifier

pipe = Pipeline([('tfidf', TfidfVectorizer(analyzer='word',ngram\_range=(1, 2))),  
 ('logreg', MultiOutputClassifier(estimator=LogisticRegression(max\_iter=10000,class\_weight='balanced',multi\_class='multinomial',))),])  
pipe.fit(X\_train, y\_train)  
  
Время обучения: 30min 41s

1. CatBoostClassifier с векторизатором TfidfVectorizer на униграммах из библиотеки sklearn. Также использовался MultiOutputClassifier

pipe = Pipeline([('tfidf', TfidfVectorizer(analyzer='word',ngram\_range=(1, 1))),('catboost', MultiOutputClassifier(estimator=CatBoostClassifier(task\_type='CPU',iterations=100,verbose=True,random\_state=42,))),])  
pipe.fit(X\_train,y\_train,catboost\_\_verbose=50)  
  
Время обучения: 2h 9min 28s

### LogisticRegression с TfidfVectorizer на униграммах

Ниже представлены результаты модели LogisticRegression с векторизатором TfidfVectorizer на униграммах.

Accuracy: 0.36987  
Hamming distance: 0.58684  
Identity (by Hamming distance): 0.98894   
  
 precision recall f1-score support  
  
 Coцсети 0.60 0.88 0.71 200  
 Авто 0.64 1.00 0.78 28  
 Автобизнес 0.41 0.97 0.57 62  
 Белоруссия 0.75 0.92 0.82 183  
 Бизнес 0.40 0.81 0.54 200  
 Бокс и ММА 0.97 0.97 0.97 200  
 Бывший СССР 0.57 0.88 0.69 200  
 Вещи 0.33 0.51 0.40 85  
 Вирусные ролики 0.50 0.89 0.64 95  
 Вкусы 0.18 0.45 0.26 11  
 Внешний вид 0.69 0.94 0.79 145  
 Гаджеты 0.49 0.89 0.63 200  
 Город 0.35 0.81 0.49 200  
 Госэкономика 0.40 0.90 0.55 200  
 Дача 0.50 0.85 0.63 200  
 Движение 0.69 0.95 0.80 167  
 Деловой климат 0.28 0.73 0.41 200  
 Деньги 0.40 0.83 0.54 200  
 Дом 0.47 0.86 0.61 200  
 Достижения 0.45 0.73 0.56 144  
 Еда 0.43 0.77 0.55 91  
 Жизнь 0.54 0.94 0.69 67  
 Закавказье 0.80 0.97 0.88 200  
 Звери 0.57 0.93 0.70 200  
 Зимние виды 0.80 0.98 0.88 200  
 Игры 0.90 0.98 0.94 200  
 Из жизни 0.65 0.82 0.72 200  
 Инструменты 0.44 0.83 0.57 142  
 Интернет 0.43 0.85 0.57 200  
 Интернет и СМИ 0.60 0.90 0.72 200  
 Искусство 0.61 0.89 0.72 200  
 История 0.45 0.80 0.57 69  
 Квартира 0.44 0.84 0.58 200  
 Киберпреступность 0.43 0.79 0.56 34  
 Кино 0.58 0.95 0.72 200  
 Книги 0.69 0.94 0.79 200  
 Конфликты 0.68 0.95 0.80 200  
 Космос 0.55 0.94 0.69 200  
 Криминал 0.37 0.87 0.52 200  
 Крым 0.53 0.92 0.67 133  
 Культура 0.51 0.78 0.61 200  
 Летние виды 0.60 0.97 0.74 200  
 Люди 0.32 0.74 0.45 200  
 Мемы 0.40 0.78 0.52 95  
 Мир 0.41 0.76 0.53 200  
 Мировой бизнес 0.33 0.75 0.46 171  
 Мнения 0.47 0.92 0.63 123  
 Молдавия 0.85 0.95 0.90 88  
 Москва 0.39 0.83 0.53 200  
 Музыка 0.72 0.95 0.82 200  
 Наука 0.49 0.89 0.63 200  
 Наука и техника 0.39 0.72 0.51 200  
 Общество 0.32 0.63 0.42 200  
 Оружие 0.57 0.93 0.71 200  
 Офис 0.37 0.79 0.51 157  
 Первая мировая 0.92 0.92 0.92 13  
 Политика 0.47 0.88 0.61 200  
Полиция и спецслужбы 0.35 0.78 0.49 200  
 Пресса 0.59 0.91 0.72 200  
 Преступная Россия 0.50 0.86 0.63 29  
 Преступность 0.47 0.92 0.62 200  
 Прибалтика 0.82 0.95 0.88 200  
 Происшествия 0.35 0.77 0.49 200  
 Путешествия 0.00 0.00 0.00 12  
 Регионы 0.32 0.74 0.45 200  
 Реклама 0.38 0.74 0.50 78  
 Россия 0.55 0.81 0.66 200  
 Рынки 0.52 0.94 0.67 200  
 Силовые структуры 0.61 0.92 0.73 200  
 Следствие и суд 0.48 0.90 0.62 200  
 События 0.40 0.77 0.52 200  
 Софт 0.43 0.88 0.57 130  
 Социальная сфера 0.14 0.21 0.17 14  
 Спорт 0.70 0.93 0.80 200  
 Средняя Азия 0.73 0.95 0.83 200  
 Стиль 0.44 0.94 0.60 200  
 ТВ и радио 0.61 0.84 0.71 200  
 Театр 0.75 0.95 0.84 200  
 Техника 0.38 0.82 0.52 200  
 Туризм 0.17 0.29 0.21 17  
 Украина 0.65 0.94 0.77 200  
 Финансы компаний 0.15 0.58 0.24 31  
 Футбол 0.67 0.98 0.80 200  
 Хоккей 0.56 0.97 0.71 183  
 Ценности 0.30 0.58 0.40 38  
 Часы 0.76 0.96 0.85 132  
 Экология 0.35 0.73 0.47 11  
 Экономика 0.51 0.85 0.64 200  
 Явления 0.42 0.84 0.56 200  
  
 micro avg 0.50 0.87 0.63 14178  
 macro avg 0.51 0.83 0.62 14178  
 weighted avg 0.53 0.87 0.65 14178  
 samples avg 0.59 0.87 0.67 14178

По многим категориям наблюдаются очень хорошие показатели recall, но низкая Accuracy (0.36987) и относительно низкое расстояние Хэмминга (0.58684) и скачущая Preсision говорит о том, что модель, всё-таки, много где предсказывает неверно. Видно, что модель плохо понимает категории, которые имеют мало значений в датасете (“Туризм”, “Социальная сфера”). Категории “Бокс и ММА” и “Белоруссия” имеют очень хорошие оценки, что говорит о хорошем обучении модели на этих тегах. Есть и такие теги, которые модель совсем не поняла (например, “Путешествия”). Уклон модели направлен в сторону Recall. На микроусреднении значение Recall равно 0.87, когда значение Precision равно 0.5. F1 в таком случае равняется 0.63. Похожесть достаточно высока, что говорит о том, что модель действительно обучилась и не делает слишком большое количество грубых ошибок.

### LogisticRegression с TfidfVectorizer на униграммах и биграммах

Ниже представлены результаты модели LogisticRegression с векторизатором TfidfVectorizer на униграммах и биграммах.

Accuracy: 0.36754  
Hamming distance: 0.58182  
Identity (by Hamming distance): 0.98904   
  
 precision recall f1-score support  
  
 Coцсети 0.55 0.88 0.68 200  
 Авто 0.73 0.96 0.83 28  
 Автобизнес 0.47 0.95 0.63 62  
 Белоруссия 0.73 0.93 0.82 183  
 Бизнес 0.43 0.80 0.56 200  
 Бокс и ММА 0.97 0.97 0.97 200  
 Бывший СССР 0.61 0.86 0.71 200  
 Вещи 0.37 0.44 0.40 85  
 Вирусные ролики 0.48 0.92 0.63 95  
 Вкусы 0.22 0.36 0.28 11  
 Внешний вид 0.65 0.94 0.77 145  
 Гаджеты 0.48 0.89 0.62 200  
 Город 0.35 0.81 0.49 200  
 Госэкономика 0.43 0.88 0.58 200  
 Дача 0.52 0.83 0.64 200  
 Движение 0.69 0.95 0.80 167  
 Деловой климат 0.31 0.73 0.43 200  
 Деньги 0.41 0.81 0.54 200  
 Дом 0.43 0.87 0.58 200  
 Достижения 0.50 0.71 0.58 144  
 Еда 0.47 0.78 0.59 91  
 Жизнь 0.59 0.93 0.72 67  
 Закавказье 0.80 0.97 0.88 200  
 Звери 0.54 0.94 0.69 200  
 Зимние виды 0.77 0.98 0.86 200  
 Игры 0.88 0.98 0.93 200  
 Из жизни 0.82 0.73 0.77 200  
 Инструменты 0.38 0.85 0.53 142  
 Интернет 0.41 0.84 0.55 200  
 Интернет и СМИ 0.58 0.90 0.71 200  
 Искусство 0.58 0.89 0.70 200  
 История 0.47 0.74 0.57 69  
 Квартира 0.43 0.86 0.57 200  
 Киберпреступность 0.29 0.79 0.43 34  
 Кино 0.55 0.94 0.69 200  
 Книги 0.68 0.94 0.79 200  
 Конфликты 0.67 0.96 0.79 200  
 Космос 0.53 0.95 0.68 200  
 Криминал 0.36 0.87 0.51 200  
 Крым 0.55 0.89 0.68 133  
 Культура 0.47 0.69 0.56 200  
 Летние виды 0.54 0.97 0.70 200  
 Люди 0.32 0.72 0.45 200  
 Мемы 0.38 0.77 0.51 95  
 Мир 0.59 0.66 0.62 200  
 Мировой бизнес 0.35 0.68 0.46 171  
 Мнения 0.46 0.93 0.62 123  
 Молдавия 0.84 0.91 0.87 88  
 Москва 0.42 0.85 0.56 200  
 Музыка 0.70 0.95 0.81 200  
 Наука 0.42 0.91 0.58 200  
 Наука и техника 0.43 0.67 0.53 200  
 Общество 0.39 0.58 0.46 200  
 Оружие 0.51 0.93 0.66 200  
 Офис 0.38 0.74 0.50 157  
 Первая мировая 1.00 0.62 0.76 13  
 Политика 0.46 0.87 0.60 200  
Полиция и спецслужбы 0.40 0.79 0.53 200  
 Пресса 0.60 0.90 0.72 200  
 Преступная Россия 0.66 0.79 0.72 29  
 Преступность 0.41 0.94 0.57 200  
 Прибалтика 0.81 0.94 0.87 200  
 Происшествия 0.27 0.81 0.40 200  
 Путешествия 0.00 0.00 0.00 12  
 Регионы 0.37 0.74 0.49 200  
 Реклама 0.40 0.77 0.53 78  
 Россия 0.64 0.80 0.71 200  
 Рынки 0.55 0.93 0.69 200  
 Силовые структуры 0.58 0.93 0.71 200  
 Следствие и суд 0.47 0.90 0.62 200  
 События 0.42 0.73 0.53 200  
 Софт 0.40 0.86 0.55 130  
 Социальная сфера 0.12 0.14 0.13 14  
 Спорт 0.61 0.92 0.73 200  
 Средняя Азия 0.74 0.95 0.83 200  
 Стиль 0.44 0.94 0.60 200  
 ТВ и радио 0.62 0.83 0.71 200  
 Театр 0.76 0.94 0.84 200  
 Техника 0.42 0.84 0.56 200  
 Туризм 0.33 0.29 0.31 17  
 Украина 0.60 0.95 0.73 200  
 Финансы компаний 0.16 0.55 0.25 31  
 Футбол 0.63 0.98 0.77 200  
 Хоккей 0.55 0.98 0.71 183  
 Ценности 0.29 0.55 0.38 38  
 Часы 0.79 0.97 0.87 132  
 Экология 0.42 0.45 0.43 11  
 Экономика 0.59 0.79 0.68 200  
 Явления 0.39 0.84 0.54 200  
  
 micro avg 0.50 0.86 0.63 14178  
 macro avg 0.52 0.81 0.62 14178  
 weighted avg 0.53 0.86 0.65 14178  
 samples avg 0.58 0.86 0.66 14178

Как и для первой модели по многим категориям наблюдаются очень хорошие показатели recall. Но, на удивление, повышения качества не произошло, и даже оно немного ухудшилось (Accuracy = 0.36754, расстояние Хэмминга = 0.58182). Для каких-то тематических категорий модель повысила качество распознавания в части Precision (например, “Туризм” (0,31), увеличился почти в два раза) До сих пор наблюдается ситуация с плохим пониманием категорий, которые имеют мало значений в датасете. Модель так же, как и первая, ничего не поняла по тематике “Путешествия”. Значения на микроусреднении по всем трём метрикам практически не поменялись.

### CatBoostClassifier с TfidfVectorizer на униграммах

Ниже представлены результаты модели CatBoostClassifier с векторизатором TfidfVectorizer на униграммах.

Accuracy: 0.47447  
Hamming distance: 0.49301  
Identity (by Hamming distance): 0.99327   
  
 precision recall f1-score support  
  
 Coцсети 0.77 0.64 0.70 200  
 Авто 0.91 0.75 0.82 28  
 Автобизнес 0.78 0.52 0.62 62  
 Белоруссия 0.81 0.84 0.83 183  
 Бизнес 0.78 0.31 0.44 200  
 Бокс и ММА 0.98 0.94 0.96 200  
 Бывший СССР 0.91 0.49 0.64 200  
 Вещи 0.67 0.05 0.09 85  
 Вирусные ролики 0.76 0.56 0.64 95  
 Вкусы 0.00 0.00 0.00 11  
 Внешний вид 0.86 0.61 0.71 145  
 Гаджеты 0.77 0.56 0.65 200  
 Город 0.55 0.28 0.37 200  
 Госэкономика 0.68 0.38 0.48 200  
 Дача 0.82 0.55 0.66 200  
 Движение 0.94 0.72 0.82 167  
 Деловой климат 0.59 0.17 0.26 200  
 Деньги 0.68 0.36 0.47 200  
 Дом 0.77 0.43 0.56 200  
 Достижения 0.66 0.24 0.36 144  
 Еда 0.56 0.31 0.40 91  
 Жизнь 0.94 0.76 0.84 67  
 Закавказье 0.91 0.86 0.88 200  
 Звери 0.72 0.56 0.63 200  
 Зимние виды 0.91 0.85 0.88 200  
 Игры 0.98 0.94 0.96 200  
 Из жизни 0.90 0.45 0.60 200  
 Инструменты 0.74 0.40 0.52 142  
 Интернет 0.65 0.35 0.46 200  
 Интернет и СМИ 0.80 0.45 0.58 200  
 Искусство 0.79 0.63 0.70 200  
 История 0.66 0.48 0.55 69  
 Квартира 0.68 0.40 0.50 200  
 Киберпреступность 0.95 0.56 0.70 34  
 Кино 0.86 0.65 0.74 200  
 Книги 0.80 0.70 0.75 200  
 Конфликты 0.88 0.72 0.79 200  
 Космос 0.68 0.59 0.64 200  
 Криминал 0.66 0.41 0.50 200  
 Крым 0.87 0.69 0.77 133  
 Культура 0.79 0.32 0.46 200  
 Летние виды 0.94 0.81 0.87 200  
 Люди 0.66 0.14 0.24 200  
 Мемы 0.88 0.54 0.67 95  
 Мир 0.74 0.14 0.24 200  
 Мировой бизнес 0.55 0.20 0.29 171  
 Мнения 0.80 0.45 0.57 123  
 Молдавия 0.93 0.91 0.92 88  
 Москва 0.73 0.42 0.54 200  
 Музыка 0.91 0.73 0.81 200  
 Наука 0.83 0.64 0.72 200  
 Наука и техника 0.78 0.33 0.46 200  
 Общество 0.83 0.12 0.22 200  
 Оружие 0.91 0.58 0.71 200  
 Офис 0.71 0.31 0.43 157  
 Первая мировая 0.60 0.46 0.52 13  
 Политика 0.81 0.38 0.52 200  
Полиция и спецслужбы 0.59 0.28 0.38 200  
 Пресса 0.76 0.53 0.62 200  
 Преступная Россия 0.74 0.59 0.65 29  
 Преступность 0.76 0.39 0.51 200  
 Прибалтика 0.90 0.83 0.86 200  
 Происшествия 0.68 0.28 0.40 200  
 Путешествия 0.00 0.00 0.00 12  
 Регионы 0.55 0.19 0.28 200  
 Реклама 0.65 0.31 0.42 78  
 Россия 0.82 0.33 0.47 200  
 Рынки 0.78 0.57 0.66 200  
 Силовые структуры 0.94 0.66 0.77 200  
 Следствие и суд 0.81 0.51 0.62 200  
 События 0.64 0.29 0.40 200  
 Софт 0.70 0.39 0.50 130  
 Социальная сфера 0.00 0.00 0.00 14  
 Спорт 0.94 0.69 0.80 200  
 Средняя Азия 0.88 0.79 0.83 200  
 Стиль 0.73 0.56 0.64 200  
 ТВ и радио 0.80 0.56 0.65 200  
 Театр 0.83 0.81 0.82 200  
 Техника 0.65 0.35 0.45 200  
 Туризм 0.00 0.00 0.00 17  
 Украина 0.90 0.71 0.79 200  
 Финансы компаний 0.50 0.06 0.11 31  
 Футбол 0.91 0.81 0.85 200  
 Хоккей 0.85 0.83 0.84 183  
 Ценности 0.62 0.13 0.22 38  
 Часы 0.90 0.86 0.88 132  
 Экология 1.00 0.27 0.43 11  
 Экономика 0.85 0.34 0.49 200  
 Явления 0.64 0.35 0.46 200  
  
 micro avg 0.81 0.51 0.63 14178  
 macro avg 0.75 0.48 0.57 14178  
 weighted avg 0.78 0.51 0.60 14178  
 samples avg 0.49 0.51 0.50 14178

Для модели бустинга на основе деревья решений значение метрики Accuracy увеличилось до 0.47447, а значение расстояния Хэмминга снизилось до 0.49301. На микроусреднении значение Recall упало до 0.51, уклон определился в сторону Precision = 0.81, как раз за счёт уменьшения Recall, F1 осталась 0.63. Модель перестала понимать категории, которые неплохо распознавались предыдущими моделями (например, “Туризм”, “Вкусы”, “Социальная сфера”).

## Нейросетевые методы машинного обучения

В качестве основных нейросетевых моделей были выбраны предобученные трансформерные языковые модели с платформы 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. Для корректной работы с каждой моделью были использованы функции, описанные ниже. Все нейросетевые модели обучались на платформе Kaggle или Colab, на предоставляемых в пользование GPU мощностях.

### Создание датасетов

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

def make\_dataset(texts, labels):  
 '''  
 Токенизация текстов и сопоставление токенов с идентификаторами  
 соответствующих им слов. Формирование PyTorch датасета  
 '''  
 input\_ids = [] # Список для токенизированных текстов  
 attention\_masks = [] # Список для масок механизма внимания  
   
 # Цикл проходится и токенизирует каждый текст  
 for seq\_to\_token in texts:  
 encoded\_dict = tokenizer.encode\_plus(  
 seq\_to\_token, # Последовательность для токенизации  
 add\_special\_tokens=True, # Добавить специальные токены в начало и в конец посл-ти  
 max\_length=338, # Максимальная длина последовательности  
 padding='max\_length', # Токен для заполнения до максимальной длины  
 return\_attention\_mask=True, # Маска механизма внимания для указания на паддинги  
 return\_tensors = 'pt', # Возвращать pytorch-тензоры  
 truncation=True # Обрезать последовательность до максимальной длины  
 )  
  
 input\_ids.append(encoded\_dict['input\_ids'])  
 attention\_masks.append(encoded\_dict['attention\_mask'])  
   
 # Конкатенация входных данных в тензоры  
 input\_ids = torch.cat(input\_ids, dim=0)  
 attention\_masks = torch.cat(attention\_masks, dim=0)  
 # Преобразование таргетов в тензоры  
 labels = torch.tensor(labels.values)  
 # Формирование датасета  
 dataset = TensorDataset(input\_ids, attention\_masks, labels)  
  
 return dataset

### Функции для обучения и валидации

Для обучения и валидации моделей были использованы функции train и validate соответственно.

def train(epoch):  
 print(f'Epoch {epoch+1}')  
 model.train()   
 fin\_targets = [] # Список для всех таргетов обучающей выборки  
 fin\_outputs = [] # Список для всех предиктов модели на обучающей выборки  
 total\_train\_loss = 0 # Функция потерь на обучении  
   
 # Цикл проходится по батчам из обучающей выборки  
 for data in train\_dataloader:  
 ids = data[0].to(device, dtype=torch.long) # Токены последовательностей из батча  
 mask = data[1].to(device, dtype=torch.long) # Маски механизма внимания последовательностей  
 targets = data[2].to(device, dtype=torch.float) # Таргеты из батча  
 res = model(ids, attention\_mask=mask, labels=targets) # В модель подаются входные тензоры и таргеты  
 loss = res['loss'] # Вычисляется значение функции потерь  
 logits = res['logits'] # Логиты предсказаний модели  
 total\_train\_loss += loss.item() # Считается функция потерь  
   
 # Таргеты и выходы модели по батчу добавляются в списки. Логиты проходят через сигмоиду  
 fin\_targets.extend(targets.cpu().detach().numpy().tolist())  
 fin\_outputs.extend(torch.sigmoid(logits).cpu().detach().numpy().tolist())  
   
 optimizer.zero\_grad() # Зануляются градиенты параметров модели  
 loss.backward() # По функции потерь рассчитываются градиенты  
 torch.nn.utils.clip\_grad\_norm\_(model.parameters(), 1.0) # Масштабируются градиенты  
 optimizer.step()   
 fin\_targets = np.array(fin\_targets)  
 fin\_outputs = np.array(fin\_outputs)  
 predictions = np.zeros(fin\_outputs.shape)  
 predictions[np.where(fin\_outputs >= 0.5)] = 1  
   
 return total\_train\_loss / len(train\_dataloader), fin\_targets, predictions  
  
def validate():  
 print(f'Validation')  
 model.eval()   
 fin\_targets = [] # Список для всех таргетов валидационной выборки  
 fin\_outputs = [] # Список для всех предиктов модели на валидационной выборки  
 total\_test\_loss = 0.0 # Функция потерь на валидации  
   
 with torch.no\_grad():  
 # Без подсчета градиентов цикл проходится по батчам  
 for data in test\_dataloader:  
 ids = data[0].to(device, dtype=torch.long) # Токены последовательностей из батча  
 mask = data[1].to(device, dtype=torch.long) # Маски механизма внимания последовательностей  
 targets = data[2].to(device, dtype=torch.float) # Таргеты из батча  
   
 res = model(ids, attention\_mask=mask, labels=targets) # В модель подаются входные тензоры и таргеты  
 loss = res['loss'] # Вычисляется значение функции потерь  
 logits = res['logits'] # Логиты предсказаний модели  
 total\_test\_loss += loss.item() # Считается функция потерь  
   
 # Таргеты и выходы модели по батчу добавляются в списки. Логиты проходят через сигмоиду  
 fin\_targets.extend(targets.cpu().detach().numpy().tolist())  
 fin\_outputs.extend(torch.sigmoid(logits).cpu().detach().numpy().tolist())  
 fin\_targets = np.array(fin\_targets)  
 fin\_outputs = np.array(fin\_outputs)  
 predictions = np.zeros(fin\_outputs.shape)  
 predictions[np.where(fin\_outputs >= 0.5)] = 1  
   
 return total\_test\_loss / len(test\_dataloader), fin\_targets, predictions

### Функции логирования и отображения метрик

Для логирования метрик был использован SummaryWriter из библиотеки PyTorch. Для реализации этого была использована функция log\_metrics. Функция log\_metrics отвечает за наполнение истории для отображения обновляющихся графиков значений метрик в процессе обучения, что осуществлялось с помощью функции plot\_learning\_curves.

def log\_metrics(history, writer, loss, targets, outputs, postfix):  
 '''  
 Расчет значений метрик и добавление их в лог обучения для отрисовки графиков.  
 Добавление значений метрик в историю обучения для отрисовки временных графиков  
 '''  
 metrics\_dict = {  
 'Loss': loss,  
 'Accuracy': metrics.accuracy\_score(targets, outputs),  
 'Hamming\_distance': hamming\_distance(targets, outputs),  
 'F1\_micro': metrics.f1\_score(targets, outputs, average='micro'),  
 'F1\_macro': metrics.f1\_score(targets, outputs, average='macro'),  
 'Recall\_micro': metrics.recall\_score(targets, outputs, average='micro'),  
 'Recall\_macro': metrics.recall\_score(targets, outputs, average='macro'),  
 'Precision\_micro': metrics.precision\_score(targets, outputs, average='micro', zero\_division=0.0),  
 'Precision\_macro': metrics.precision\_score(targets, outputs, average='macro', zero\_division=0.0)  
 }  
   
 for metric, value in metrics\_dict.items():  
 if not 'macro' in metric:  
 history[metric][postfix].append(value)  
 writer.add\_scalar(f'{metric}/{postfix}', value, epoch)  
  
def plot\_learning\_curves(history):  
 '''  
 Отрисовка обновляющихся графиков значений метрик,  
 для отслеживания в процессе обучения  
 '''  
 fig = plt.figure(figsize=(20, 10))  
   
 for i, metric in enumerate(history.keys(), 1):  
 plt.subplot(2,3,i)  
 plt.title(metric, fontsize=15)  
 plt.plot(range(1, epoch+2), history[metric]['train'], label='train')  
 plt.plot(range(1, epoch+2), history[metric]['val'], label='val')  
 plt.xticks(range(1, epoch+2))  
 if i > 3:  
 plt.xlabel('epoch', fontsize=15)  
 plt.legend()  
  
 plt.show()

### Инициализация

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

device = torch.device('cuda' if torch.cuda.is\_available() else 'cpu')  
  
selected\_model = 'cointegrated/rubert-tiny2'  
tokenizer = AutoTokenizer.from\_pretrained(selected\_model)  
  
BATCH\_SIZE = 32  
EPOCHS = 20  
EARLY\_STOP = 3  
OPT = torch.optim.NAdam  
LEARNING\_RATE = 3e-5  
EPSILON = 1e-8  
SCHEDULER = False  
SAMPLE = False

# Инициализируется предобученная модель  
model = AutoModelForSequenceClassification.from\_pretrained(  
 selected\_model,  
 problem\_type='multi\_label\_classification', # Решается задача многоклассовой классификации. Функция потерь BCEWithLogitsLoss  
 num\_labels=y\_test.shape[1], # Число классов  
 output\_attentions = False, # Модель не выдает результаты работы механизма внимания  
 output\_hidden\_states = False # Модель не выдает скрытые состояния  
)  
model.to(device)  
  
# Инициализируется оптимизатор  
optimizer = OPT(  
 model.parameters(),  
 lr=LEARNING\_RATE,  
 eps=EPSILON)  
  
# Инициализируется шедулер  
scheduler = get\_linear\_schedule\_with\_warmup(  
 optimizer,  
 num\_warmup\_steps=0,  
 num\_training\_steps=len(train\_dataloader) \* EPOCHS)  
  
# Инициализируется инструмент логирования  
writer = SummaryWriter(  
 comment= '-' + selected\_model.replace('/', '-'))

### Обучение модели

В ходе обучения была задействован метод ранней остановки. Метрикой для ранней остановки была выбрана F1-measure. Обучение моделей происходило по следующему алгоритму:

# Переменная для хранения лучшей метрики для ранней остановки  
best\_f1\_val = 0  
# Переменная для отсчета количества эпох с момента лучшей метрики  
epochs\_since\_best = 0  
# Словарь для хранения истории метрик  
history = defaultdict(lambda: defaultdict(list))  
  
for epoch in range(EPOCHS):  
  
 avg\_train\_loss, targets, outputs = train(epoch) # Обучение модели на одной эпохе  
 log\_metrics(history, writer, avg\_train\_loss, targets, outputs, 'train') # Логирование метрик  
  
 avg\_val\_loss, targets, outputs = validate() # Предсказания модели на вал. выборке  
 log\_metrics(history, writer, avg\_val\_loss, targets, outputs, 'val') # Логирование метрик  
  
 clear\_output()  
 # Отрисовка кривых обучения  
 plot\_learning\_curves(history)  
 torch.save(model, model\_path + 'models\_files\\' + f"{selected\_model.replace('/', '-')}\_{epoch}.pt")  
 # Расчет micro f1-score на валидационной выборке  
 f1\_val = metrics.f1\_score(targets, outputs, average='micro')  
 # Если метрика лучше предыдущей лучшей, то сохраняется модель  
 if f1\_val > best\_f1\_val:  
 best\_f1\_val = f1\_val  
 torch.save(model, model\_path + 'models\_files\\' + f"{selected\_model.replace('/', '-')}\_{epoch}.pt")  
 epochs\_since\_best = 0  
 # В противном случае идет отсчет эпох до ранней остановки  
 else:  
 epochs\_since\_best += 1  
 print('Best epoch:', epoch, '\nBest F1-score:',best\_f1\_val, '\n')  
 if epochs\_since\_best == EARLY\_STOP:  
 break  
  
writer.flush()  
writer.close()

### Испытания

Для подбора гиперпараметров каждой модели испытания проводились на 25% данных от обучающей выборки. Скорость обучения выбиралась из диапазона 2\*10^-5 – 4\*10^-5. В ходе испытаний рассматривались оптимизаторы Adam, NAdam, AdamW библиотеки Pytorch. Размер батча выбирался по характеристикам моделей и изменялся в диапазоне от 8 до 128. Использование шедулера не дало ожидаемого результата, отчего из дальнейшего исследования он был исключен.

### Модели

#### **Трансформер ruBERT tiny2 (Cointegrated)**

Первый трансформер для исследования - ruBERT tiny2 от Cointegrated, версия небольшого энкодера для русского языка, основанного на модели BERT, с 29.4 млн. параметров. Для обучения использовались следующие гиперпараметры:

device = torch.device('cuda' if torch.cuda.is\_available() else 'cpu')  
  
selected\_model = 'cointegrated/rubert-tiny2'  
tokenizer = AutoTokenizer.from\_pretrained(selected\_model)  
  
BATCH\_SIZE = 32  
EPOCHS = 20  
EARLY\_STOP = 3  
OPT = torch.optim.NAdam  
LEARNING\_RATE = 3e-5  
EPSILON = 1e-8  
SCHEDULER = False  
SAMPLE = False  
  
Время обучения: 59min 18s

Обучение модели продлилось 15 эпох, при этом лучшей метрики F1 модель достигла уже на 12 эпохе.

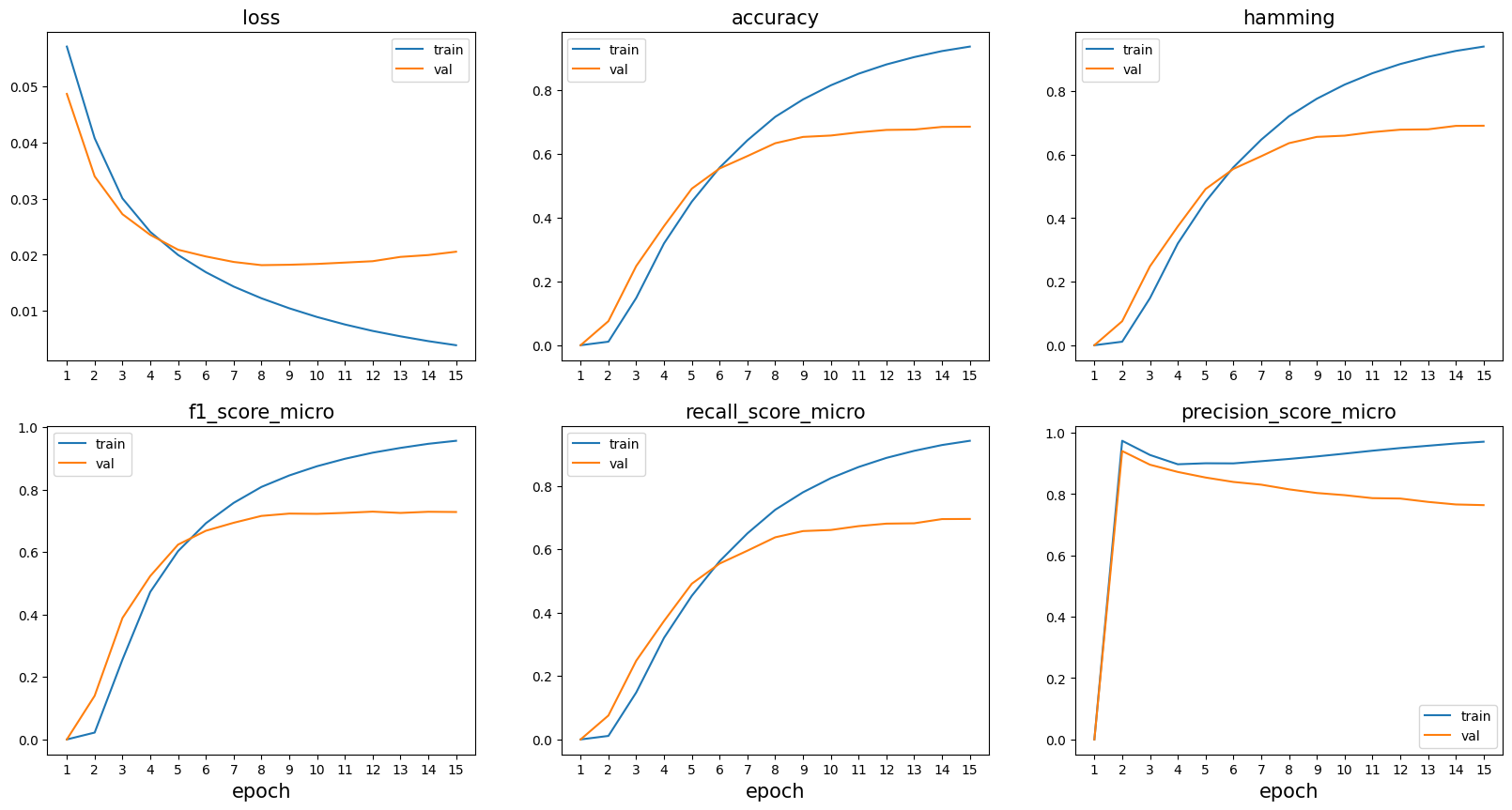


Рис. 5. Кривые обучения ruBERT tiny2

Accuracy: 0.68557  
Hamming distance: 0.69075  
Identity (by Hamming distance): 0.99429   
  
 precision recall f1-score support  
  
 Coцсети 0.82 0.74 0.78 200  
 Авто 0.96 0.96 0.96 28  
 Автобизнес 0.68 0.73 0.70 62  
 Белоруссия 0.89 0.86 0.88 183  
 Бизнес 0.72 0.53 0.61 200  
 Бокс и ММА 0.98 0.95 0.97 200  
 Бывший СССР 0.77 0.82 0.79 200  
 Вещи 0.50 0.15 0.23 85  
 Вирусные ролики 0.76 0.72 0.74 95  
 Вкусы 0.00 0.00 0.00 11  
 Внешний вид 0.81 0.90 0.85 145  
 Гаджеты 0.74 0.73 0.74 200  
 Город 0.54 0.53 0.53 200  
 Госэкономика 0.73 0.56 0.63 200  
 Дача 0.73 0.70 0.72 200  
 Движение 0.88 0.89 0.88 167  
 Деловой климат 0.51 0.42 0.46 200  
 Деньги 0.58 0.56 0.57 200  
 Дом 0.75 0.65 0.70 200  
 Достижения 0.58 0.35 0.43 144  
 Еда 0.56 0.55 0.56 91  
 Жизнь 0.83 0.81 0.82 67  
 Закавказье 0.90 0.94 0.92 200  
 Звери 0.77 0.69 0.72 200  
 Зимние виды 0.96 0.94 0.95 200  
 Игры 0.97 0.96 0.97 200  
 Из жизни 0.82 0.71 0.76 200  
 Инструменты 0.72 0.64 0.68 142  
 Интернет 0.72 0.62 0.67 200  
 Интернет и СМИ 0.82 0.76 0.79 200  
 Искусство 0.80 0.79 0.79 200  
 История 0.71 0.68 0.70 69  
 Квартира 0.71 0.60 0.65 200  
 Киберпреступность 0.87 0.59 0.70 34  
 Кино 0.93 0.85 0.89 200  
 Книги 0.86 0.85 0.86 200  
 Конфликты 0.83 0.83 0.83 200  
 Космос 0.76 0.71 0.73 200  
 Криминал 0.58 0.65 0.61 200  
 Крым 0.78 0.88 0.83 133  
 Культура 0.78 0.78 0.78 200  
 Летние виды 0.93 0.87 0.90 200  
 Люди 0.54 0.45 0.49 200  
 Мемы 0.67 0.59 0.63 95  
 Мир 0.65 0.55 0.59 200  
 Мировой бизнес 0.58 0.35 0.44 171  
 Мнения 0.72 0.59 0.65 123  
 Молдавия 0.94 0.99 0.96 88  
 Москва 0.63 0.66 0.64 200  
 Музыка 0.88 0.88 0.88 200  
 Наука 0.85 0.62 0.72 200  
 Наука и техника 0.67 0.64 0.65 200  
 Общество 0.50 0.28 0.36 200  
 Оружие 0.84 0.83 0.84 200  
 Офис 0.59 0.43 0.50 157  
 Первая мировая 0.00 0.00 0.00 13  
 Политика 0.75 0.54 0.62 200  
Полиция и спецслужбы 0.66 0.47 0.55 200  
 Пресса 0.76 0.68 0.72 200  
 Преступная Россия 0.77 0.59 0.67 29  
 Преступность 0.72 0.79 0.75 200  
 Прибалтика 0.92 0.95 0.94 200  
 Происшествия 0.63 0.55 0.58 200  
 Путешествия 0.00 0.00 0.00 12  
 Регионы 0.52 0.49 0.51 200  
 Реклама 0.62 0.33 0.43 78  
 Россия 0.74 0.72 0.73 200  
 Рынки 0.79 0.73 0.76 200  
 Силовые структуры 0.91 0.77 0.83 200  
 Следствие и суд 0.75 0.61 0.68 200  
 События 0.59 0.56 0.57 200  
 Софт 0.64 0.57 0.60 130  
 Социальная сфера 0.00 0.00 0.00 14  
 Спорт 0.91 0.95 0.93 200  
 Средняя Азия 0.93 0.94 0.94 200  
 Стиль 0.69 0.78 0.73 200  
 ТВ и радио 0.80 0.69 0.74 200  
 Театр 0.87 0.90 0.88 200  
 Техника 0.61 0.59 0.60 200  
 Туризм 0.00 0.00 0.00 17  
 Украина 0.89 0.91 0.90 200  
 Финансы компаний 0.00 0.00 0.00 31  
 Футбол 0.95 0.94 0.95 200  
 Хоккей 0.86 0.92 0.89 183  
 Ценности 0.47 0.21 0.29 38  
 Часы 0.93 0.94 0.93 132  
 Экология 0.00 0.00 0.00 11  
 Экономика 0.75 0.69 0.72 200  
 Явления 0.69 0.54 0.61 200  
  
 micro avg 0.76 0.70 0.73 14178  
 macro avg 0.69 0.64 0.66 14178  
 weighted avg 0.75 0.70 0.72 14178  
 samples avg 0.69 0.70 0.69 14178

По многим категориям наблюдаются чрезвычайно хорошие показатели по всем метрикам. Повышение произошло у Accuracy до 0.68557, у расстояния Хэмминга до 0.69075. К категориям, у которых мало записей в датасете, модель не смогла отнести ни одного представителя. Рост Precision виден практически по всем категориям, а потому уклон идет в сторону него. На микроусреднении Recall упал, по сравнению с моделями логистической регрессии, до 0.70, но зато Precision вырос с 0.50 до 0.76, что привело к повышению F1 с 0.63 до 0.73.

#### **Трансформер ruBERT base (AI forever)**

Трансформер **ruBERT base** от AI forever, полновесный энкодер для русского языка с количеством параметров равным 178 млн. Для обучения использовались следующие гиперпараметры:

device = torch.device('cuda' if torch.cuda.is\_available() else 'cpu')  
  
selected\_model = 'ai-forever/rubert-base'  
tokenizer = AutoTokenizer.from\_pretrained(selected\_model)  
  
BATCH\_SIZE = 32  
EPOCHS = 20  
EARLY\_STOP = 2  
OPT = torch.optim.NAdam  
LEARNING\_RATE = 3e-5  
EPSILON = 1e-8  
SAMPLE = False  
SCHEDULER = False  
Время обучения: 6h 4min 56s

Обучение модели продлилось 9 эпох, при этом лучшей метрики F1 модель достигла уже на 7 эпохе.

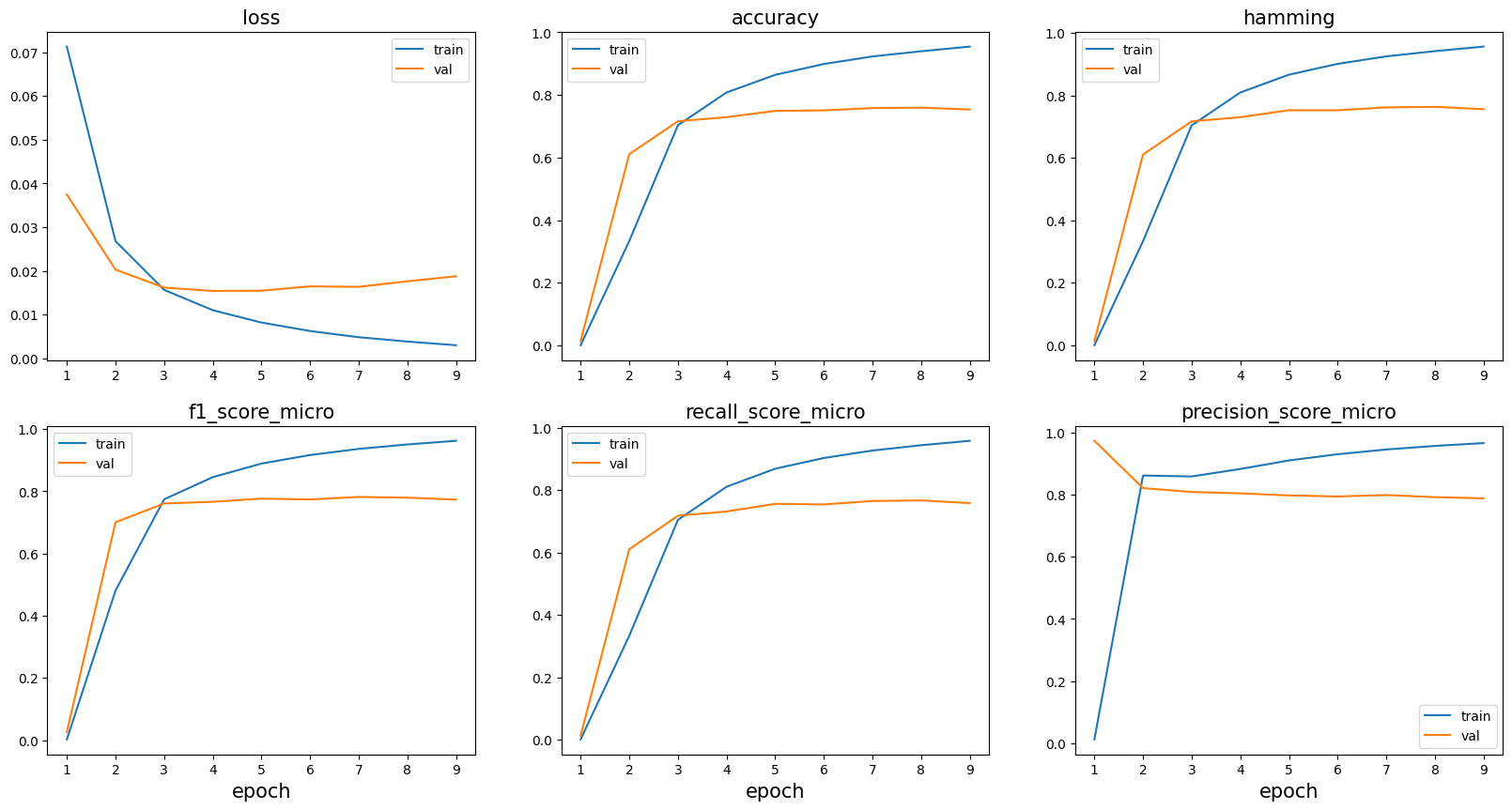


Рис. 6. Кривые обучения ruBERT base

Accuracy: 0.75801  
Hamming distance: 0.76153  
Identity (by Hamming distance): 0.9953   
  
 precision recall f1-score support  
  
 Coцсети 0.84 0.76 0.80 200  
 Авто 0.88 1.00 0.93 28  
 Автобизнес 0.81 0.76 0.78 62  
 Белоруссия 0.93 0.91 0.92 183  
 Бизнес 0.74 0.69 0.72 200  
 Бокс и ММА 0.99 0.97 0.98 200  
 Бывший СССР 0.82 0.86 0.84 200  
 Вещи 0.62 0.33 0.43 85  
 Вирусные ролики 0.79 0.75 0.77 95  
 Вкусы 0.00 0.00 0.00 11  
 Внешний вид 0.84 0.90 0.87 145  
 Гаджеты 0.82 0.67 0.73 200  
 Город 0.59 0.58 0.59 200  
 Госэкономика 0.74 0.72 0.73 200  
 Дача 0.79 0.76 0.77 200  
 Движение 0.84 0.92 0.88 167  
 Деловой климат 0.55 0.56 0.56 200  
 Деньги 0.69 0.61 0.65 200  
 Дом 0.85 0.67 0.75 200  
 Достижения 0.70 0.60 0.64 144  
 Еда 0.69 0.63 0.66 91  
 Жизнь 0.83 0.93 0.87 67  
 Закавказье 0.92 0.94 0.93 200  
 Звери 0.81 0.79 0.80 200  
 Зимние виды 0.96 0.96 0.96 200  
 Игры 0.92 0.98 0.95 200  
 Из жизни 0.87 0.80 0.83 200  
 Инструменты 0.83 0.65 0.73 142  
 Интернет 0.81 0.62 0.70 200  
 Интернет и СМИ 0.87 0.86 0.87 200  
 Искусство 0.89 0.81 0.85 200  
 История 0.84 0.77 0.80 69  
 Квартира 0.66 0.78 0.71 200  
 Киберпреступность 0.78 0.74 0.76 34  
 Кино 0.95 0.91 0.93 200  
 Книги 0.86 0.93 0.89 200  
 Конфликты 0.86 0.81 0.83 200  
 Космос 0.76 0.83 0.79 200  
 Криминал 0.70 0.51 0.59 200  
 Крым 0.91 0.86 0.88 133  
 Культура 0.88 0.84 0.86 200  
 Летние виды 0.93 0.92 0.92 200  
 Люди 0.65 0.59 0.62 200  
 Мемы 0.70 0.75 0.72 95  
 Мир 0.73 0.65 0.69 200  
 Мировой бизнес 0.69 0.48 0.57 171  
 Мнения 0.74 0.73 0.73 123  
 Молдавия 0.98 1.00 0.99 88  
 Москва 0.79 0.70 0.74 200  
 Музыка 0.86 0.96 0.91 200  
 Наука 0.83 0.80 0.81 200  
 Наука и техника 0.78 0.76 0.77 200  
 Общество 0.62 0.51 0.56 200  
 Оружие 0.86 0.91 0.89 200  
 Офис 0.63 0.50 0.56 157  
 Первая мировая 1.00 0.69 0.82 13  
 Политика 0.74 0.69 0.71 200  
Полиция и спецслужбы 0.66 0.66 0.66 200  
 Пресса 0.71 0.82 0.76 200  
 Преступная Россия 0.84 0.72 0.78 29  
 Преступность 0.78 0.81 0.80 200  
 Прибалтика 0.95 0.95 0.95 200  
 Происшествия 0.70 0.69 0.70 200  
 Путешествия 0.00 0.00 0.00 12  
 Регионы 0.66 0.55 0.60 200  
 Реклама 0.65 0.67 0.66 78  
 Россия 0.88 0.73 0.80 200  
 Рынки 0.83 0.72 0.77 200  
 Силовые структуры 0.95 0.84 0.89 200  
 Следствие и суд 0.76 0.80 0.78 200  
 События 0.56 0.65 0.60 200  
 Софт 0.72 0.62 0.67 130  
 Социальная сфера 0.00 0.00 0.00 14  
 Спорт 0.93 0.96 0.95 200  
 Средняя Азия 0.93 0.94 0.94 200  
 Стиль 0.76 0.78 0.77 200  
 ТВ и радио 0.83 0.71 0.77 200  
 Театр 0.92 0.88 0.90 200  
 Техника 0.60 0.71 0.65 200  
 Туризм 0.38 0.18 0.24 17  
 Украина 0.95 0.91 0.93 200  
 Финансы компаний 0.33 0.03 0.06 31  
 Футбол 0.97 0.92 0.94 200  
 Хоккей 0.89 0.93 0.91 183  
 Ценности 0.60 0.47 0.53 38  
 Часы 0.94 0.89 0.92 132  
 Экология 0.14 0.09 0.11 11  
 Экономика 0.75 0.84 0.80 200  
 Явления 0.67 0.64 0.65 200  
  
 micro avg 0.80 0.77 0.78 14178  
 macro avg 0.75 0.72 0.73 14178  
 weighted avg 0.80 0.77 0.78 14178  
 samples avg 0.76 0.77 0.76 14178

Практически по всем категориям относительно ruBERT tiny наблюдается прирост для всех метрик. Повышение произошло у Accuracy до 0.75801, у расстояния Хэмминга до 0.76153. Проблемными категориями для модели остались “Социальная сфера”, “Путешествия”, “Вкусы”. Уклон остался также в сторону Precision. На микроусреднении Recall изменился до 0.77, Precision поднялся до 0.80, что повлекло рост F1 до 0.78.

#### **Трансформер DistilBERT (Geotrend)**

Трансформер **DistilBERT base ru cased** от Geotrend, версия мультиязычного энкодера для русского языка c количеством параметров 54.5 млн. Для обучения использовались следующие гиперпараметры:

device = torch.device('cuda' if torch.cuda.is\_available() else 'cpu')  
  
selected\_model = 'Geotrend/distilbert-base-ru-cased'  
tokenizer = AutoTokenizer.from\_pretrained(selected\_model)  
  
BATCH\_SIZE = 32  
EPOCHS = 20  
EARLY\_STOP = 3  
OPT = torch.optim.NAdam  
LEARNING\_RATE = 3e-5  
EPSILON = 1e-8  
SCHEDULER = False  
SAMPLE = False  
  
Время обучения: 4h 21min 7s

Обучение модели продлилось 13 эпох, при этом лучшей метрики F1 модель достигла уже на 10 эпохе.

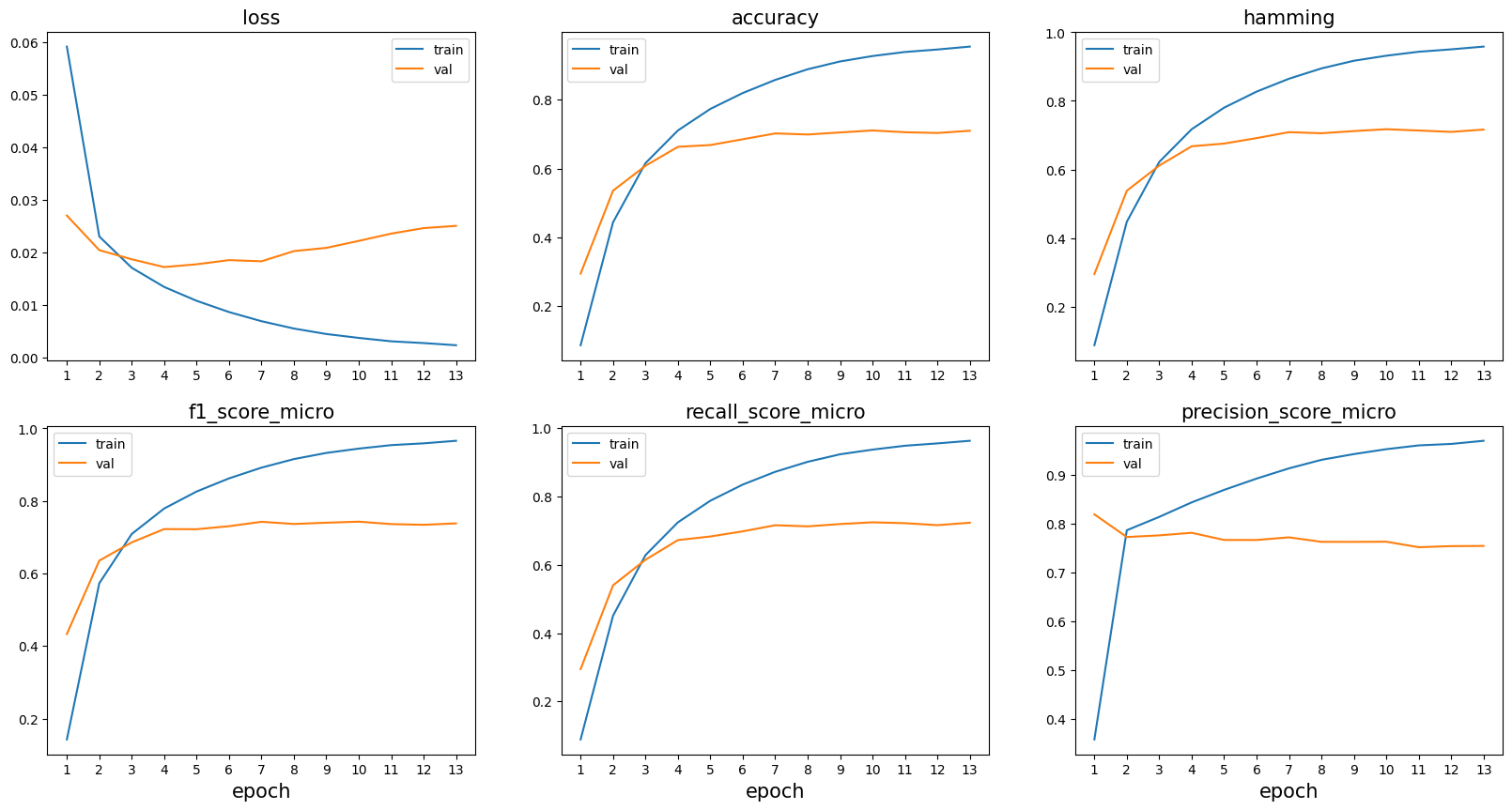


Рис. 7. Кривые обучения DistilBERT

Accuracy: 0.70976  
Hamming distance: 0.71629  
Identity (by Hamming distance): 0.99435   
  
 precision recall f1-score support  
  
 Coцсети 0.77 0.69 0.73 200  
 Авто 0.93 0.89 0.91 28  
 Автобизнес 0.71 0.74 0.72 62  
 Белоруссия 0.91 0.90 0.90 183  
 Бизнес 0.76 0.45 0.56 200  
 Бокс и ММА 0.98 0.96 0.97 200  
 Бывший СССР 0.82 0.84 0.83 200  
 Вещи 0.45 0.38 0.41 85  
 Вирусные ролики 0.76 0.76 0.76 95  
 Вкусы 0.50 0.09 0.15 11  
 Внешний вид 0.86 0.83 0.85 145  
 Гаджеты 0.67 0.71 0.69 200  
 Город 0.64 0.44 0.52 200  
 Госэкономика 0.72 0.61 0.66 200  
 Дача 0.71 0.69 0.70 200  
 Движение 0.81 0.92 0.86 167  
 Деловой климат 0.49 0.39 0.43 200  
 Деньги 0.66 0.52 0.58 200  
 Дом 0.73 0.77 0.75 200  
 Достижения 0.69 0.62 0.65 144  
 Еда 0.51 0.57 0.54 91  
 Жизнь 0.89 0.87 0.88 67  
 Закавказье 0.88 0.94 0.91 200  
 Звери 0.77 0.72 0.75 200  
 Зимние виды 0.96 0.93 0.94 200  
 Игры 0.94 0.97 0.96 200  
 Из жизни 0.80 0.69 0.74 200  
 Инструменты 0.63 0.70 0.66 142  
 Интернет 0.73 0.61 0.67 200  
 Интернет и СМИ 0.73 0.81 0.77 200  
 Искусство 0.89 0.76 0.82 200  
 История 0.70 0.68 0.69 69  
 Квартира 0.63 0.72 0.67 200  
 Киберпреступность 0.86 0.56 0.68 34  
 Кино 0.91 0.91 0.91 200  
 Книги 0.91 0.79 0.84 200  
 Конфликты 0.83 0.84 0.84 200  
 Космос 0.84 0.73 0.78 200  
 Криминал 0.60 0.60 0.60 200  
 Крым 0.82 0.90 0.86 133  
 Культура 0.83 0.77 0.80 200  
 Летние виды 0.94 0.88 0.91 200  
 Люди 0.54 0.55 0.55 200  
 Мемы 0.71 0.73 0.72 95  
 Мир 0.60 0.58 0.59 200  
 Мировой бизнес 0.44 0.53 0.48 171  
 Мнения 0.72 0.66 0.69 123  
 Молдавия 0.94 1.00 0.97 88  
 Москва 0.60 0.77 0.67 200  
 Музыка 0.89 0.89 0.89 200  
 Наука 0.87 0.64 0.73 200  
 Наука и техника 0.71 0.73 0.72 200  
 Общество 0.55 0.44 0.49 200  
 Оружие 0.78 0.87 0.82 200  
 Офис 0.77 0.42 0.54 157  
 Первая мировая 0.67 0.77 0.71 13  
 Политика 0.61 0.65 0.63 200  
Полиция и спецслужбы 0.56 0.58 0.57 200  
 Пресса 0.77 0.75 0.76 200  
 Преступная Россия 0.71 0.86 0.78 29  
 Преступность 0.72 0.71 0.72 200  
 Прибалтика 0.94 0.92 0.93 200  
 Происшествия 0.79 0.51 0.62 200  
 Путешествия 0.00 0.00 0.00 12  
 Регионы 0.56 0.60 0.58 200  
 Реклама 0.57 0.50 0.53 78  
 Россия 0.74 0.76 0.75 200  
 Рынки 0.78 0.80 0.79 200  
 Силовые структуры 0.94 0.64 0.76 200  
 Следствие и суд 0.70 0.73 0.72 200  
 События 0.66 0.55 0.60 200  
 Софт 0.49 0.75 0.59 130  
 Социальная сфера 0.33 0.07 0.12 14  
 Спорт 0.92 0.92 0.92 200  
 Средняя Азия 0.89 0.93 0.91 200  
 Стиль 0.78 0.74 0.76 200  
 ТВ и радио 0.83 0.66 0.73 200  
 Театр 0.88 0.88 0.88 200  
 Техника 0.61 0.60 0.61 200  
 Туризм 0.29 0.24 0.26 17  
 Украина 0.90 0.93 0.91 200  
 Финансы компаний 0.50 0.19 0.28 31  
 Футбол 0.96 0.93 0.94 200  
 Хоккей 0.82 0.95 0.88 183  
 Ценности 0.54 0.37 0.44 38  
 Часы 0.83 0.98 0.90 132  
 Экология 0.25 0.09 0.13 11  
 Экономика 0.77 0.78 0.77 200  
 Явления 0.68 0.53 0.60 200  
  
 micro avg 0.75 0.72 0.74 14178  
 macro avg 0.72 0.68 0.69 14178  
 weighted avg 0.76 0.72 0.73 14178  
 samples avg 0.72 0.72 0.72 14178

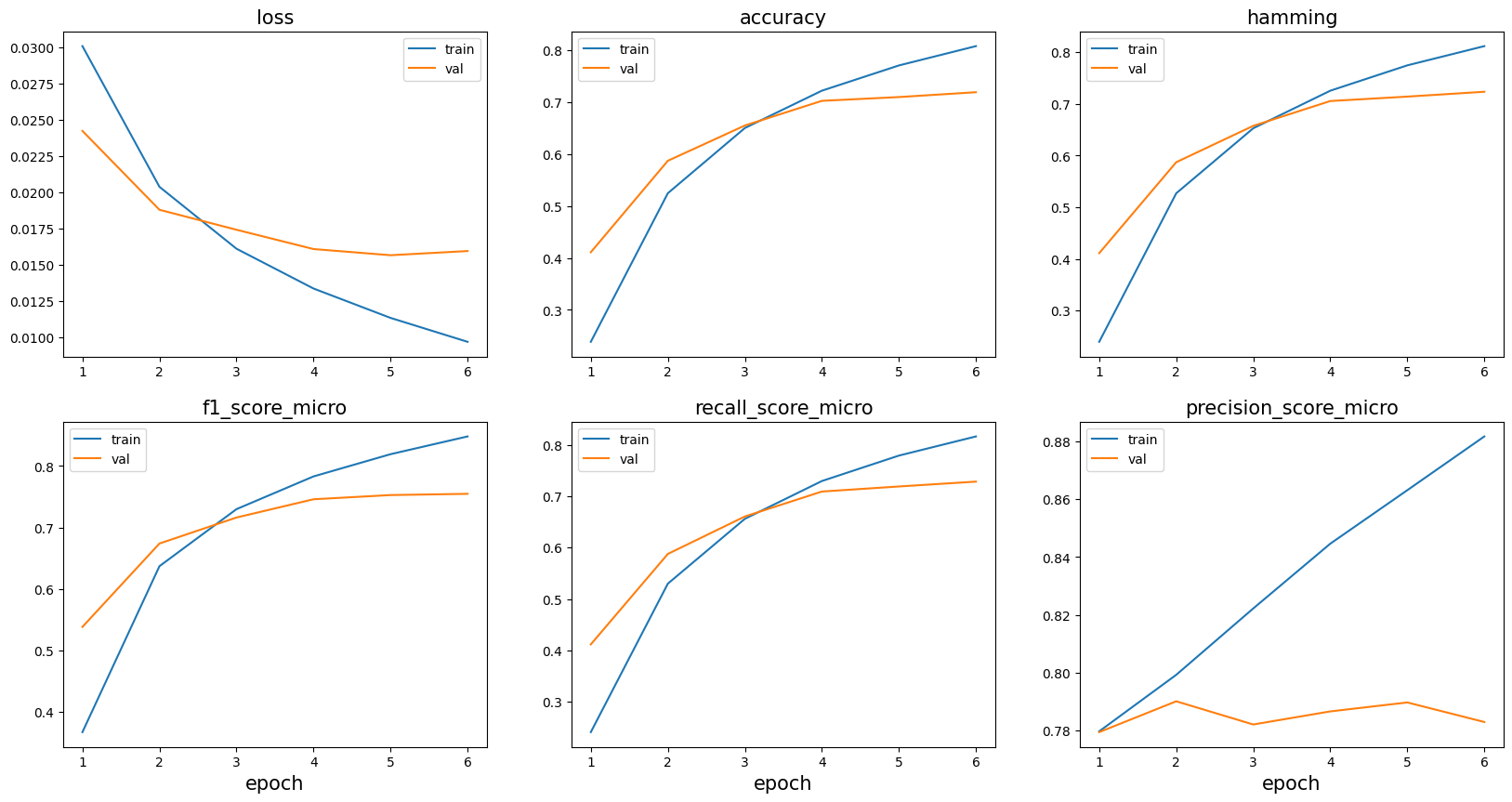
По всем своим метрикам DistilBERT занимает промежуточное положение между ruBERT tiny2 и ruBERT base. Модель DistilBERT лучше отработала проблемные категории, осталась нераспознанной только категория “Путешествия”. Практически по всем категориям относительно ruBert tiny наблюдается прирост для всех метрик. Значение Accuracy равно 0.70976, расстояние Хэмминга равно 0.71629. Уклон остался также в сторону Precision. На микроусреднении Recall = 0.72, Precision = 0.75, F1 = 0.74.

#### **Трансформер XLM-RoBERTa (FacebookAI)**

Трансформер XLM-RoBERTa base от FacebookAI – это единственная рассматриваемая здесь мультиязычная модель с количеством параметров 279 млн. Модель обучалась в два этапа, так как время сессии предоставляемых мощностей вышло. Для обучения использовались следующие гиперпараметры:

device = torch.device('cuda' if torch.cuda.is\_available() else 'cpu')  
  
selected\_model = 'FacebookAI/xlm-roberta-base'  
tokenizer = AutoTokenizer.from\_pretrained(selected\_model)  
  
BATCH\_SIZE = 8  
EPOCHS = 20  
EARLY\_STOP = 2  
OPT = torch.optim.Adam  
LEARNING\_RATE = 2e-5  
EPSILON = 1e-8  
SCHEDULER = False  
SAMPLE = False  
Время обучения: 7h 46min 15s + 5h 12min 3s = 12h 58min 18s

Обучение модели продлилось 6 эпох в первую итерацию и 6 эпох во вторую. Итого 12 эпох, при этом лучшей метрики F1 модель достигла уже на 10 эпохе.



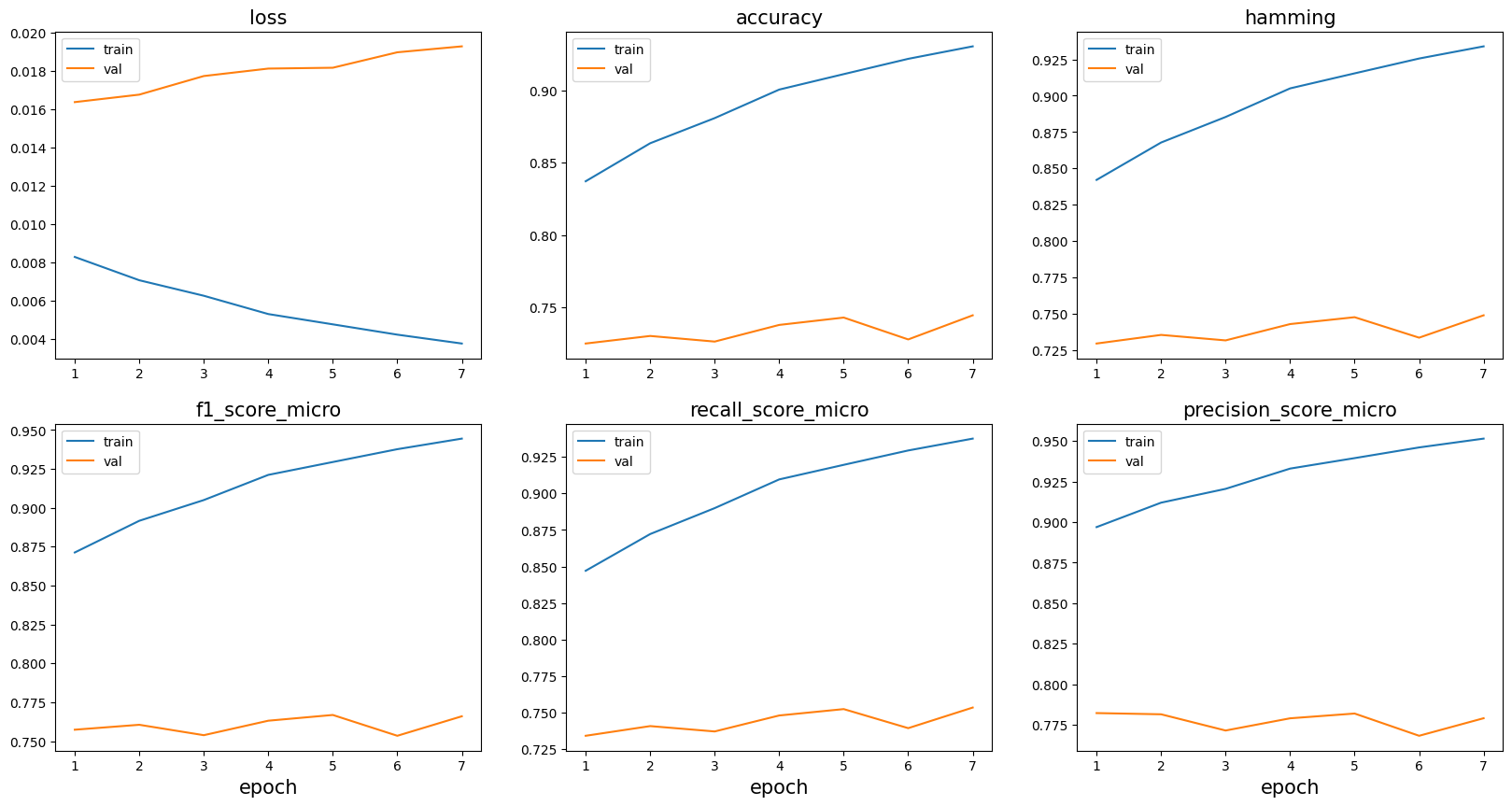


Рис. 8. Кривые обучения XLM-RoBERTa

Accuracy: 0.74312  
Hamming distance: 0.74774  
Identity (by Hamming distance): 0.99496   
  
 precision recall f1-score support  
  
 Coцсети 0.78 0.80 0.79 200  
 Авто 0.96 0.96 0.96 28  
 Автобизнес 0.52 0.87 0.65 62  
 Белоруссия 0.89 0.93 0.91 183  
 Бизнес 0.71 0.67 0.69 200  
 Бокс и ММА 0.98 0.96 0.97 200  
 Бывший СССР 0.88 0.81 0.84 200  
 Вещи 0.52 0.41 0.46 85  
 Вирусные ролики 0.75 0.84 0.80 95  
 Вкусы 0.00 0.00 0.00 11  
 Внешний вид 0.80 0.91 0.85 145  
 Гаджеты 0.75 0.71 0.73 200  
 Город 0.58 0.59 0.58 200  
 Госэкономика 0.64 0.67 0.65 200  
 Дача 0.76 0.74 0.75 200  
 Движение 0.85 0.95 0.90 167  
 Деловой климат 0.51 0.39 0.44 200  
 Деньги 0.57 0.70 0.63 200  
 Дом 0.73 0.77 0.75 200  
 Достижения 0.66 0.60 0.63 144  
 Еда 0.63 0.69 0.66 91  
 Жизнь 0.82 0.84 0.83 67  
 Закавказье 0.90 0.94 0.92 200  
 Звери 0.79 0.76 0.77 200  
 Зимние виды 0.96 0.92 0.94 200  
 Игры 0.98 0.96 0.97 200  
 Из жизни 0.76 0.86 0.81 200  
 Инструменты 0.74 0.80 0.77 142  
 Интернет 0.77 0.69 0.72 200  
 Интернет и СМИ 0.81 0.89 0.85 200  
 Искусство 0.84 0.85 0.85 200  
 История 0.68 0.72 0.70 69  
 Квартира 0.73 0.71 0.72 200  
 Киберпреступность 0.73 0.79 0.76 34  
 Кино 0.97 0.86 0.91 200  
 Книги 0.87 0.88 0.87 200  
 Конфликты 0.86 0.80 0.83 200  
 Космос 0.73 0.81 0.77 200  
 Криминал 0.58 0.72 0.65 200  
 Крым 0.82 0.87 0.85 133  
 Культура 0.81 0.92 0.86 200  
 Летние виды 0.96 0.85 0.90 200  
 Люди 0.63 0.62 0.63 200  
 Мемы 0.69 0.65 0.67 95  
 Мир 0.85 0.56 0.68 200  
 Мировой бизнес 0.60 0.38 0.47 171  
 Мнения 0.75 0.78 0.76 123  
 Молдавия 0.98 0.99 0.98 88  
 Москва 0.72 0.70 0.71 200  
 Музыка 0.94 0.91 0.92 200  
 Наука 0.81 0.75 0.78 200  
 Наука и техника 0.69 0.79 0.73 200  
 Общество 0.57 0.41 0.48 200  
 Оружие 0.89 0.82 0.86 200  
 Офис 0.59 0.56 0.58 157  
 Первая мировая 0.76 1.00 0.87 13  
 Политика 0.70 0.69 0.70 200  
Полиция и спецслужбы 0.72 0.47 0.56 200  
 Пресса 0.75 0.81 0.78 200  
 Преступная Россия 0.80 0.83 0.81 29  
 Преступность 0.72 0.79 0.75 200  
 Прибалтика 0.94 0.96 0.95 200  
 Происшествия 0.87 0.52 0.65 200  
 Путешествия 0.25 0.08 0.12 12  
 Регионы 0.63 0.52 0.57 200  
 Реклама 0.51 0.62 0.55 78  
 Россия 0.86 0.77 0.81 200  
 Рынки 0.87 0.58 0.70 200  
 Силовые структуры 0.95 0.76 0.84 200  
 Следствие и суд 0.79 0.70 0.74 200  
 События 0.73 0.49 0.59 200  
 Софт 0.67 0.72 0.69 130  
 Социальная сфера 0.20 0.07 0.11 14  
 Спорт 0.91 0.97 0.94 200  
 Средняя Азия 0.91 0.98 0.94 200  
 Стиль 0.75 0.83 0.79 200  
 ТВ и радио 0.83 0.76 0.79 200  
 Театр 0.90 0.90 0.90 200  
 Техника 0.76 0.44 0.56 200  
 Туризм 0.55 0.35 0.43 17  
 Украина 0.93 0.88 0.90 200  
 Финансы компаний 0.55 0.19 0.29 31  
 Футбол 0.97 0.94 0.95 200  
 Хоккей 0.80 0.95 0.87 183  
 Ценности 0.63 0.45 0.52 38  
 Часы 0.93 0.92 0.92 132  
 Экология 0.50 0.45 0.48 11  
 Экономика 0.81 0.81 0.81 200  
 Явления 0.72 0.60 0.66 200  
  
 micro avg 0.78 0.75 0.77 14178  
 macro avg 0.75 0.72 0.73 14178  
 weighted avg 0.78 0.75 0.76 14178  
 samples avg 0.75 0.75 0.75 14178

Значение Accuracy установилось в 0.74312, расстояние Хэмминга установилось в 0.74774. По всем своим метрикам модель не показала глобальных изменений, заняв промежуточное положение между ruBERT base и DistilBERT. Стоит отметить, что увеличение параметров всё же дало такой эффект, что самая проблемная нераспознанная категория “Путешествия” была определена со следующими характеристиками: Precision = 0.25, Recall = 0.08 F1 = 0.12. На микроусреднении Recall = 0.75, Precision = 0.78, F1 = 0.77.

# Сравнение моделей

## Сравнение нейросетевых моделей

По всем метрикам лидирующую позицию занимает ruBERT base. Модель ruBERT tiny2 является достаточно неплохой, если учесть тот факт, что она по всем метрикам стартует с заметно более низкой базы, но в ходе обучения постепенно приближается к показателям остальных архитектур. Показатели моделей XLM-RoBERTa и DistilBERT достаточно близки, что свидетельствует в пользу эффективности DistilBERT, у которого количество параметров меньше почти в 6 раз.

### ruBERT base

Accuracy: 0.75801  
Hamming distance: 0.76153  
 precision recall f1-score  
 micro avg 0.80 0.77 0.78

### XLM-RoBERTa

Accuracy: 0.74312  
Hamming distance: 0.74774  
 precision recall f1-score  
 micro avg 0.78 0.75 0.77

### DistilBERT

Accuracy: 0.70976  
Hamming distance: 0.71629  
 precision recall f1-score  
 micro avg 0.75 0.72 0.74

### ruBERT tiny2

Accuracy: 0.68557  
Hamming distance: 0.69075  
 precision recall f1-score  
 micro avg 0.76 0.70 0.73

Стоит подсветить, что минимизация функции потерь достигается раньше, чем максимизация метрик F1, Hamming distance, Recall, что хорошо видно по графикам. Хотя минимальная функция потерь соответствует максимальному Accuracy, эта метрика не несёт большого смысла в рамках поставленной задачи многоклассовой классификации. Более нужными выглядят F1 и Hamming distance, максимизация которых происходит за счет быстрого роста Recall на фоне постепенного и периодического понижения Precision – это также общее поведение для всех моделей. Также на графиках хорошо видится схожесть кривых метрик F1 и Hamming distance, что говорит о близости смысла метрик.

## Сравнение классических и нейросетевых моделей

Сравнение классических и нейросетевых моделей в сводной таблице основных характеристик и метрик.

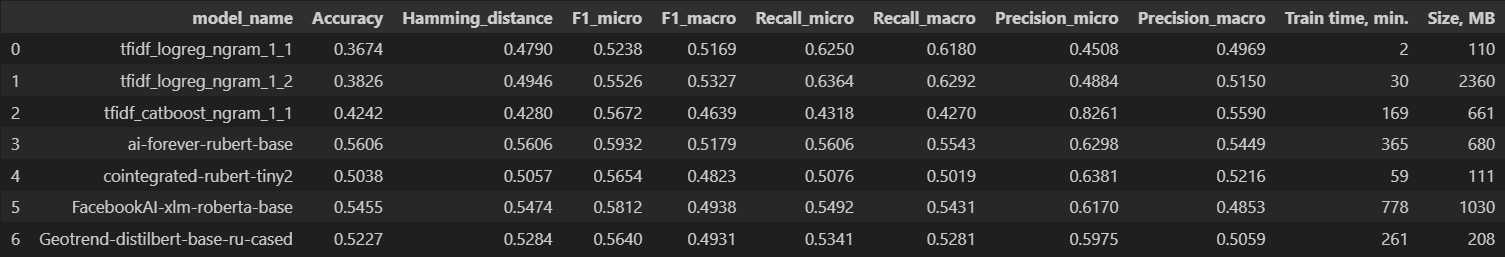


Рис. 9. Сравнение моделей

При сравнении моделей стоит опираться также на вес и время обучения. Классические методы (TFIDF и логистическая регрессия, деревья решений) являются самым быстрым решением, которое требует относительно немного места для моделей на униграммах – всего 110 МБ для модели логистической регрессии. Но такой подход не даёт высоких значений метрик. Хорошим вариантом для улучшения результатов является ансамблирование моделей.

При тонкой настройке нейросетевые модели способны показывать высокое качество. Здесь следует не забывать о том, что количество параметров, а соответственно вес модели и время на обучение, растут быстрее значений метрик. Поэтому для успешного использования нейросетевых моделей необходимо обладать достаточными ресурсами, либо мириться с ограничениями и увеличением времени работы. Хорошим вариантом для улучшения качества нейросетевых моделей может стать аугментация данных.

# Примеры

Для сравнения оцениваемых моделей было решено протестировать их на данных из датасета “example.csv”, собранного из данных исходного датасета с переработанной фильтрацией. Использовался скрипт для получения датафрейма с первой колонкой включающей лемматизированный текст, со второй колонкой, включающей правильные номера категорий/тегов, с остальными колонками, включающими в себя предсказания от каждой модели для лемматизированного текста из первой колонки.

df\_ex = pd.read\_csv('example.csv')  
df\_ex.fillna(0, inplace = True)  
columns\_lst = df\_ex.columns[1:]  
df\_final\_example = df\_ex.drop(columns=columns\_lst)  
models\_list\_comparison = ['ai-forever-rubert-base.pt', 'cointegrated-rubert-tiny2.pt', 'FacebookAI-xlm-roberta-base.pt', 'Geotrend-distilbert-base-ru-cased.pt', 'model\_tfidf\_catboost\_ngram\_1\_1.pkl','model\_tfidf\_logreg\_ngram\_1\_1.pkl', 'model\_tfidf\_logreg\_ngram\_1\_2.pkl']  
  
for k in models\_list\_comparison:  
 print(k[:-3])  
 try:  
 df\_pred = pd.read\_csv('df\_pred\_' + k[:-3] + '.csv')  
 except:  
 df\_pred = pd.read\_csv('df\_pred\_' + k[6:-4] + '.csv')  
 indices\_pred\_list = []  
 indices\_true\_list = []  
 for elem in range(len(df\_pred)):  
 indices\_pred = set(np.where(df\_pred.to\_numpy()[elem])[0])  
 indices\_true = set(np.where(df\_ex[columns\_lst].to\_numpy()[elem])[0])  
 indices\_pred\_list.append(indices\_pred)  
 indices\_true\_list.append(indices\_true)  
 df\_final\_example[f'id\_true\_{k[:-3]}'] = indices\_true\_list  
 df\_final\_example[f'id\_pred\_{k[:-3]}'] = indices\_pred\_list  
columns\_final\_ex = df\_final\_example.columns

Также использовалась формула comparison\_result для вывода результатов предсказаний по одному лемматизированному тексту для разных моделей

def comparison\_result(st\_ind,end\_ind,columns\_lst, columns\_final\_ex):  
 for col in range(1,15,2):  
 st\_col, end\_col = columns\_final\_ex[col], columns\_final\_ex[col+1]  
 print(st\_col, end\_col)  
 for i in range(len(df\_final\_example[st\_ind:end\_ind].loc[:,['text\_lemma',st\_col,end\_col]])):  
 text\_lemm = df\_final\_example[st\_ind:end\_ind].loc[:,['text\_lemma',st\_col,end\_col]].iloc[i][0]  
 true\_tag = columns\_lst[list(df\_final\_example[st\_ind:end\_ind].loc[:,['text\_lemma',st\_col,end\_col]].iloc[i][1])[0]]  
 print('model:', st\_col[8:])  
 print('text\_lemmatize:', text\_lemm)  
 print('true\_tag:', true\_tag)  
 try:  
 pred\_tag = list(df\_final\_example[st\_ind:end\_ind].loc[:,['text\_lemma',st\_col,end\_col]].iloc[i][2])  
 list\_pred = []  
 for tag in pred\_tag:  
 list\_pred.append(columns\_lst[tag])  
 print('pred\_tag:',', '.join(list\_pred))  
 except:  
 pred\_tag = ''  
 print('pred\_tag:', pred\_tag)

comparison\_result(170,171,columns\_lst, columns\_final\_ex)  
# вывод переработан для сокращения  
>>>  
'''  
Президент России Владимир Путин своим указом произвел в генералы 33 полковника различных ведомств — от Министерства обороны до Военной прокуратуры. Соответствующий указ в среду, 12 декабря, опубликован на официальном портале правовой информации. Кроме того, еще 16 генералам присвоены очередные воинские звания. В частности, в Минобороны России появился один новый генерал-полковник и один полный адмирал, восемь генерал-лейтенантов и два приравненных к ним вице-адмирала, 10 генерал-майоров и один контр-адмирал. В МЧС появиллось два генерал-полковника, четыре генерал-лейтенанта и семь генерал-майоров. В свою очередь, Росгвардия пополнилась одним генерал-полковником, тремя генерал-лейтенантами и девятью генерал-майорами. В Военной прокуратуре появился один генерал-лейтенант юстиции и три генерал-майора. Наконец, Федеральная служба исполнения наказаний (ФСИН) получила двух генерал-лейтенантов и четырех генерал-майоров. Так, генерал-лейтенант Евгений Устинов, командующий войсками Центрального военного округа стал генерал-полковником, а командующий Балтийским флотом вице-адмирал Александр Носатов отныне — полный адмирал. Командующий Уральским округом Росгвардии генерал-лейтенант Александр Попов получил на погоны третью шитую звезду и стал генерал-полковником. В основном генеральские звания присвоены офицерам, чей «потолок по должности» с недавнего времени увеличен — в течение ноября-декабря президент внес изменения в положения почти о всех силовых структурах, увеличив число генеральских должностей.  
'''  
  
model: ai-forever-rubert-base  
true\_tag: Полиция и спецслужбы  
pred\_tag: Полиция и спецслужбы  
  
model: cointegrated-rubert-tiny2  
true\_tag: Полиция и спецслужбы  
pred\_tag: Полиция и спецслужбы  
  
model: FacebookAI-xlm-roberta-base  
true\_tag: Полиция и спецслужбы  
pred\_tag: Полиция и спецслужбы  
  
model: Geotrend-distilbert-base-ru-cased  
true\_tag: Полиция и спецслужбы  
pred\_tag: Полиция и спецслужбы  
  
model: model\_tfidf\_catboost\_ngram\_1\_1.  
true\_tag: Полиция и спецслужбы  
pred\_tag: Полиция и спецслужбы  
  
model: model\_tfidf\_logreg\_ngram\_1\_1.  
true\_tag: Полиция и спецслужбы  
pred\_tag: Полиция и спецслужбы, Россия, Оружие  
  
model: model\_tfidf\_logreg\_ngram\_1\_2.  
true\_tag: Полиция и спецслужбы  
pred\_tag: Полиция и спецслужбы, Россия, Общество, Оружие

comparison\_result(263,264,columns\_lst, columns\_final\_ex)  
# вывод переработан для сокращения  
>>>  
'''  
Улицу бутиков в «Барвиха Luxury Village» украсили к Новому году по мотивам сказки Гофмана «Щелкунчик и Мышиный Король». Об этом сообщается в пресс-релизе, присланном в редакцию «Ленты.ру» в четверг, 13 декабря. В центре внимания — красавица Мари, главная героиня сказки. Фигура главного положительного героя — Щелкунчика — установлена у праздничной елки возле Barvikha Hotel & Spa, украшенной бутафорскими конфетами, игрушками, бантами и коробками с подарками. Его сказочный антогонист — семиголовый Мышиный Король — венчает стилизованный торте праздничной карусели. Карусель «оккупировала» армия Мышиного короля, охраняющая золотые орехи и бронзовые пушки. Одна из самых романтичных инсталляций — па-де-де, которое танцуют Фея Драже и принц Оршад: их силуэты мягко подсвечены. На улице бутиков звучит музыка из знаменитого балета Петра Ильича Чайковского: барабанный бой, фанфары, дивертисмент, «Вальс цветов» и заключительное лирическое адажио. Ранее в ноябре московский универмаг ЦУМ открыл традиционный рождественский базар. Фасад универмага «перевязали» красными лентами с бантами, а стены украсили рождественскими звездами. В атриуме ЦУМа возвышается десятиметровая новогодняя елка с золотистыми украшениями, а галереи этажей декорированы бантами, разноцветными шарами, звездами и подарочными коробками.  
'''  
  
model: ai-forever-rubert-base  
true\_tag: Явления  
pred\_tag: Явления  
  
model: cointegrated-rubert-tiny2  
true\_tag: Явления  
pred\_tag: Явления  
  
model: FacebookAI-xlm-roberta-base  
true\_tag: Явления  
pred\_tag: Инструменты  
  
model: Geotrend-distilbert-base-ru-cased  
true\_tag: Явления  
pred\_tag: Явления  
  
model: model\_tfidf\_catboost\_ngram\_1\_1.  
true\_tag: Явления  
pred\_tag: Явления  
  
model: model\_tfidf\_logreg\_ngram\_1\_1.  
true\_tag: Явления  
pred\_tag: Явления, Инструменты  
  
model: model\_tfidf\_logreg\_ngram\_1\_2.  
true\_tag: Явления  
pred\_tag: Явления, Инструменты

На примерах хорошо видно разницу в результатах работы алгоритмов. Логистическая регрессия на tfidf-векторах хорошо определяет не только целевые категории, но и смежные к ним, отсюда должно быть большее количество ложноположительных предсказаний и более низкий Precision. Нейросети более точечны в своих предсказаниях, а потому лучше подстраиваются под имеющиеся данные.

# Список литературы

1. [Multi Label Model Evaluation](file:///D:\netology_diplom\final\text_files\kaggle.com\code\kmkarakaya\multi-label-model-evaluation)
2. [Hamming score](file:///D:\netology_diplom\final\text_files\wiki.cloudfactory.com\docs\mp-wiki\metrics\hamming-score)
3. [Лемматизируй это быстрее (PyMorphy2, PyMystem3 и немного магии)](file:///D:\netology_diplom\final\text_files\habr.com\ru\articles\503420)
4. [Github. iterative-stratification](file:///D:\netology_diplom\final\text_files\github.com\trent-b\iterative-stratification%3ftab=readme-ov-file)
5. [Multi Label Binary Classification with CatBoost](file:///D:\netology_diplom\final\text_files\github.com\catboost\tutorials\blob\master\classification\multilabel_classification_tutorial.ipynb)
6. [Hugging Face](file:///D:\netology_diplom\final\text_files\huggingface.co)
7. [Все, что нужно знать об ALBERT, RoBERTa и DistilBERT](file:///D:\netology_diplom\final\text_files\habr.com\ru\articles\680986)
8. [Исходный датасет (800 000 строк)](https://www.kaggle.com/datasets/yutkin/corpus-of-russian-news-articles-from-lenta/data)
9. [Обработанный датасет (70889 строк)](https://www.kaggle.com/datasets/trofimovpaul/news-datasets-train-test)