нетология

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

Репозиторий: Дипломная работа. Github

Датасеты: Исходный датасет (800 000 строк), Обработанный датасет (70889 строк)

2025

Содержание

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** 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)

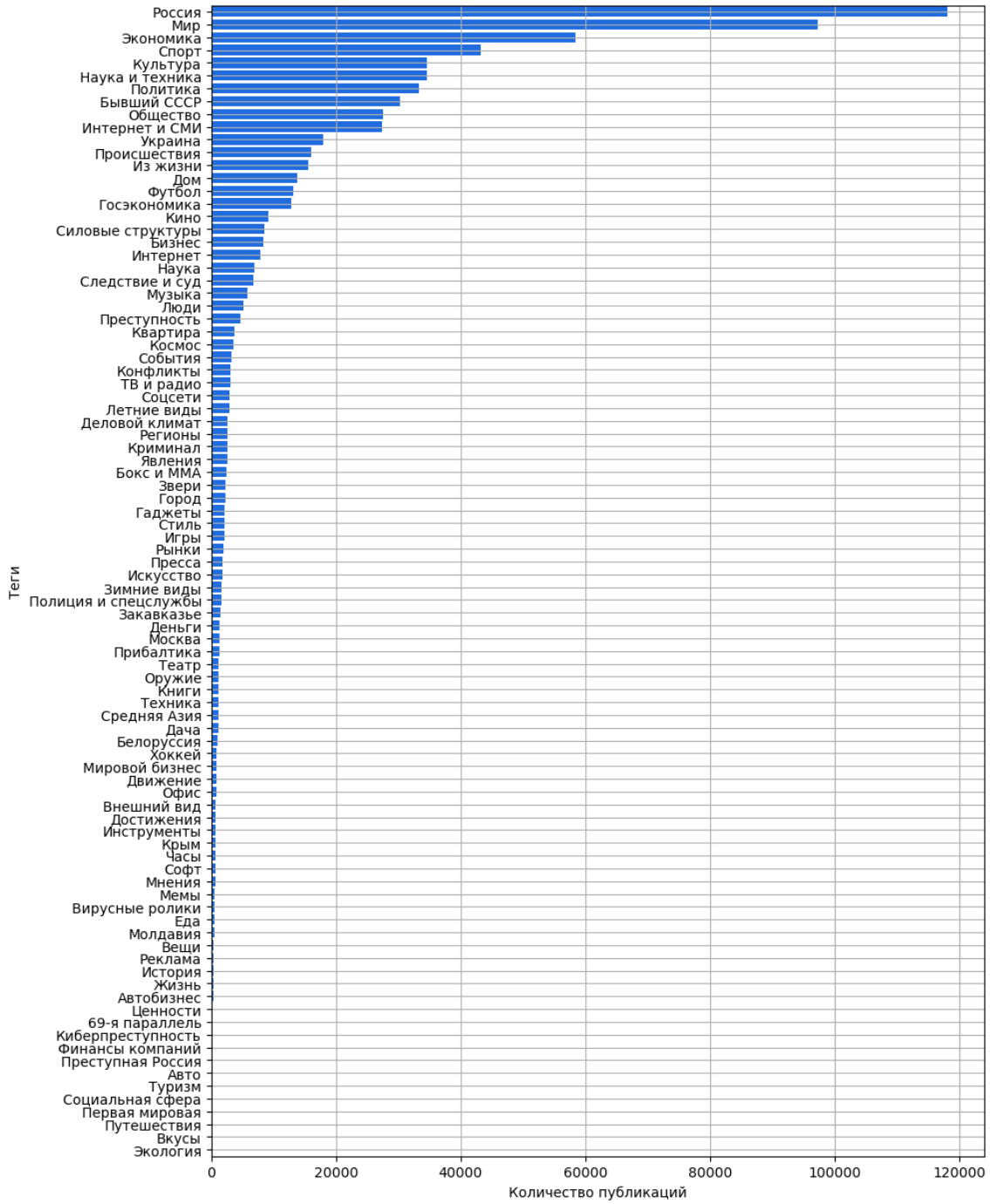

Рис. 1. Сравнение количества новостных публикаций по тегам

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.

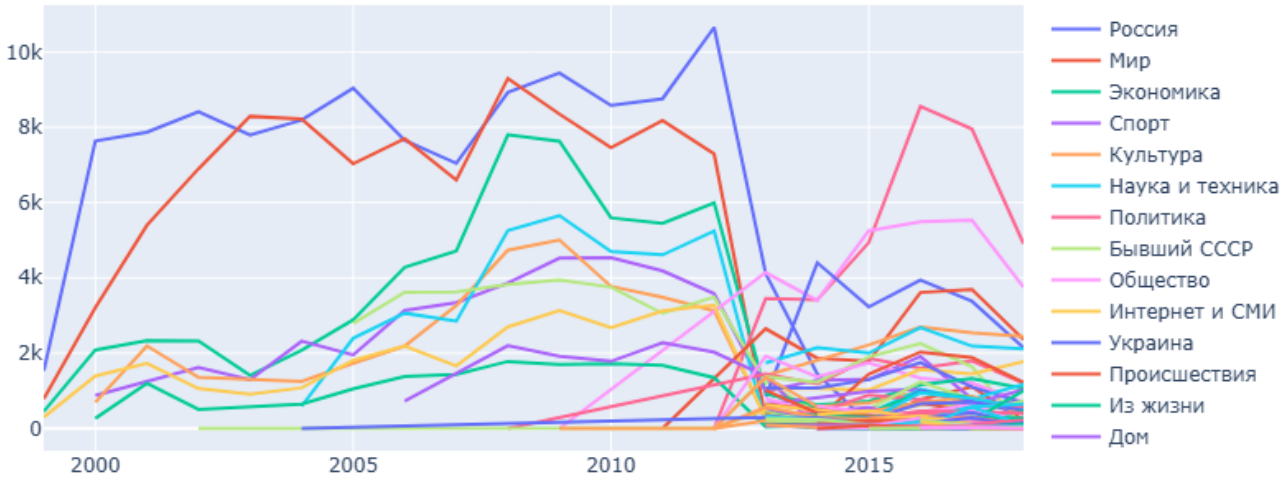

Рис. 2. Сравнение количества публикаций по годам для каждой тематической категории (тега)

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

2. 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}.c
sv', 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')
```

	text	Соцсети	Авто	Автобизнес	Белоруссия	Бизнес	Бокс и ММА	Бывший СССР	Вещи	Вирусные ролики	...	Туризм	Украина	Финансы компаний	Футбол	Хоккей	Ценности	Часы	Экология	Экономика	Явления
0	Вице-премьер Владислав Сурков заявил, что не и...	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	Владислав Цыплухин, занимавший пост руководите...	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	Сервис микроблогов Twitter откроет представите...	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	Ежемесячная аудитория фотосервиса Instagram со...	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	Пользователи Facebook загрузили на Новый год в...	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
70884	Американская супермодель Джиджи Хадид разработ...	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
70885	Американский предприниматель Марк Фарезе (Mark...	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
70886	Новой темой ежегодного Met Gala в музее Метроп...	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
70887	Французская актриса Марион Котийар приняла уча...	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
70888	Британский модельер Пол Смит стал приглашенным...	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1

70889 rows × 90 columns

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.

$$Accuracy = (TP + TN)/(TP + TN + FP + FN)$$

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.

$$Precision = TP/(TP + FP)$$

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.

$$Recall = TP/(TP + FN)$$

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.

$$F1 = (2 * Recall * Precision)/(Recall + Precision) =$$

$$= TP/\big(TP + 0.5 * (FP + FN)\big)$$

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)

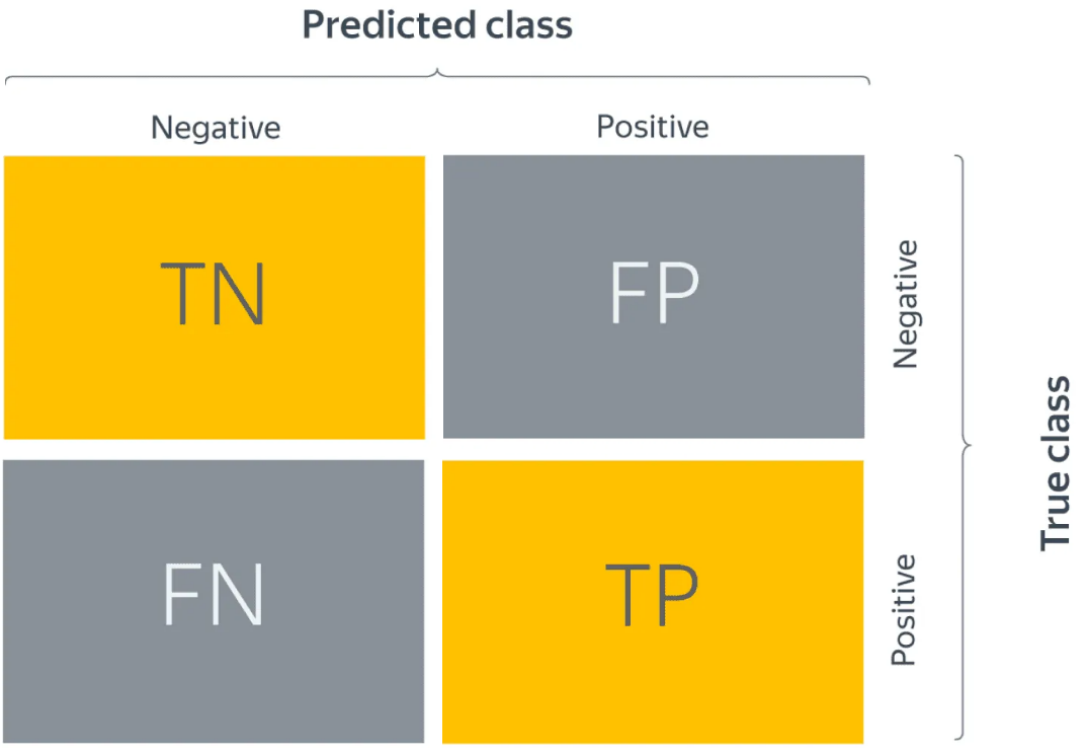

Рис. 4. Схема матрицы несоответствия

```
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 должн
о
    работать уравнение: figsize_h = 6 * ncols)
    figsize_h - высота полотна для вывода сетки матриц несоответствия (для корректного отображения сетки должн
о
    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_pr
ed = 1)
    #   или о грубой ошибке при предсказании (y_true = 1, y_pred = 0) категории, имеет смысл выводить информац
ию 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)} c')
```

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.

$$Identity = 1 - d_h(obj1, obj2)/length(obj1)$$

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.u
nion(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', mult
i_class='multinomial',))),])
pipe.fit(X_train, y_train)

Время обучения: 2min 32s
```

2. 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
```

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

```
pipe = Pipeline([('tfidf', TfidfVectorizer(analyzer='word',ngram_range=(1, 1))),('catboost', MultiOutputClassi
fier(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
Соцсети	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 Precision 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
Соцсети	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
Соцсети	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 последовательност
ей
            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 таргет
ы
            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. выбо
рке
    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.

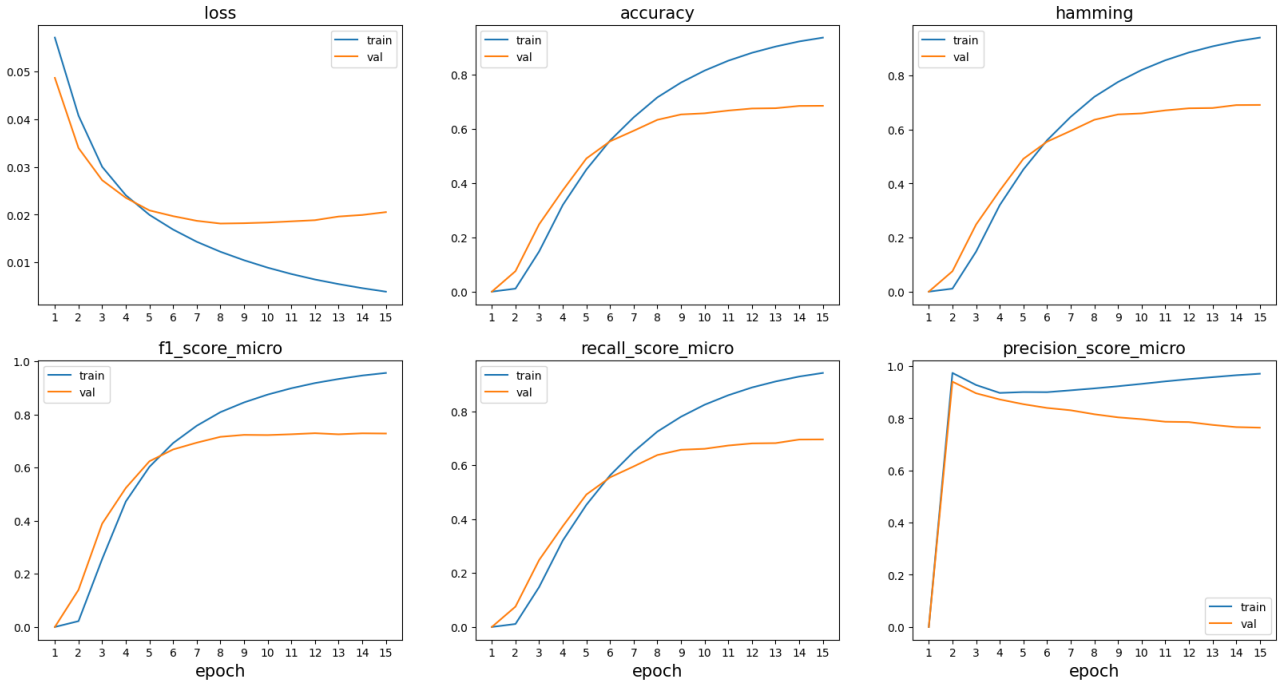

Рис. 5. Кривые обучения ruBERT tiny2

Accuracy: 0.68557
Hamming distance: 0.69075
Identity (by Hamming distance): 0.99429

	precision	recall	f1-score	support
Соцсети	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.

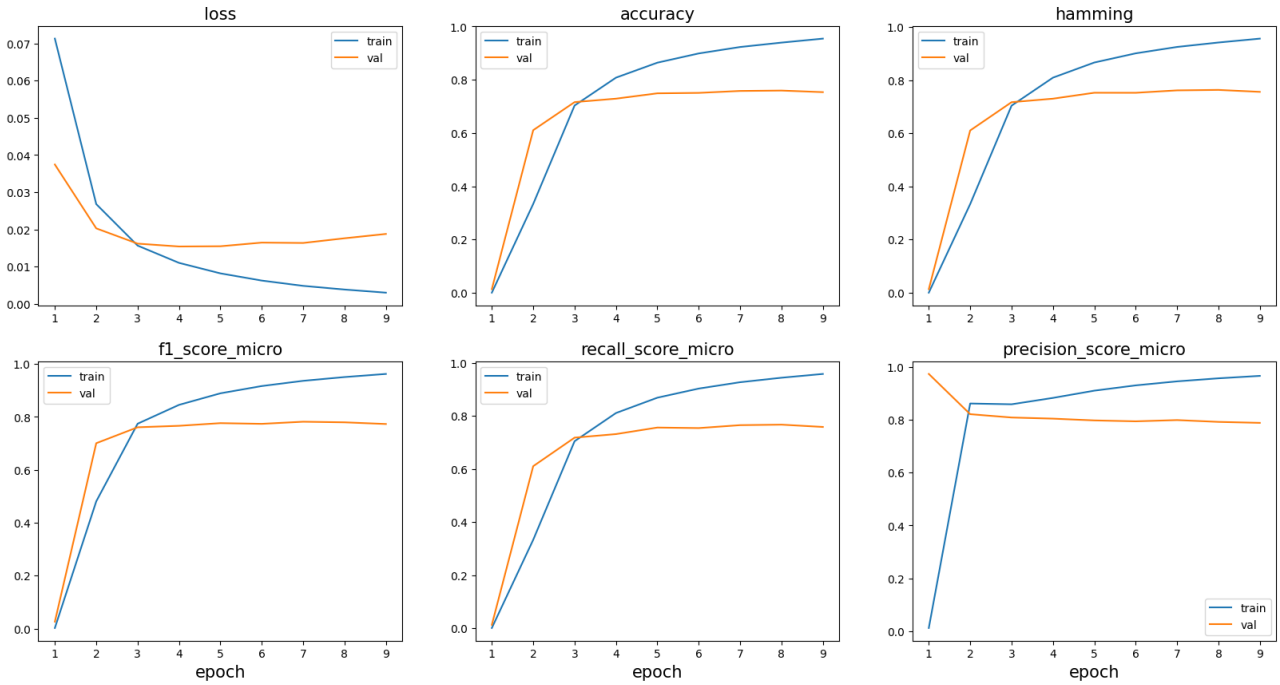

Рис. 6. Кривые обучения ruBERT base

```
Accuracy: 0.75801
Hamming distance: 0.76153
Identity (by Hamming distance): 0.9953
```

	precision	recall	f1-score	support
Соцсети	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 с 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.

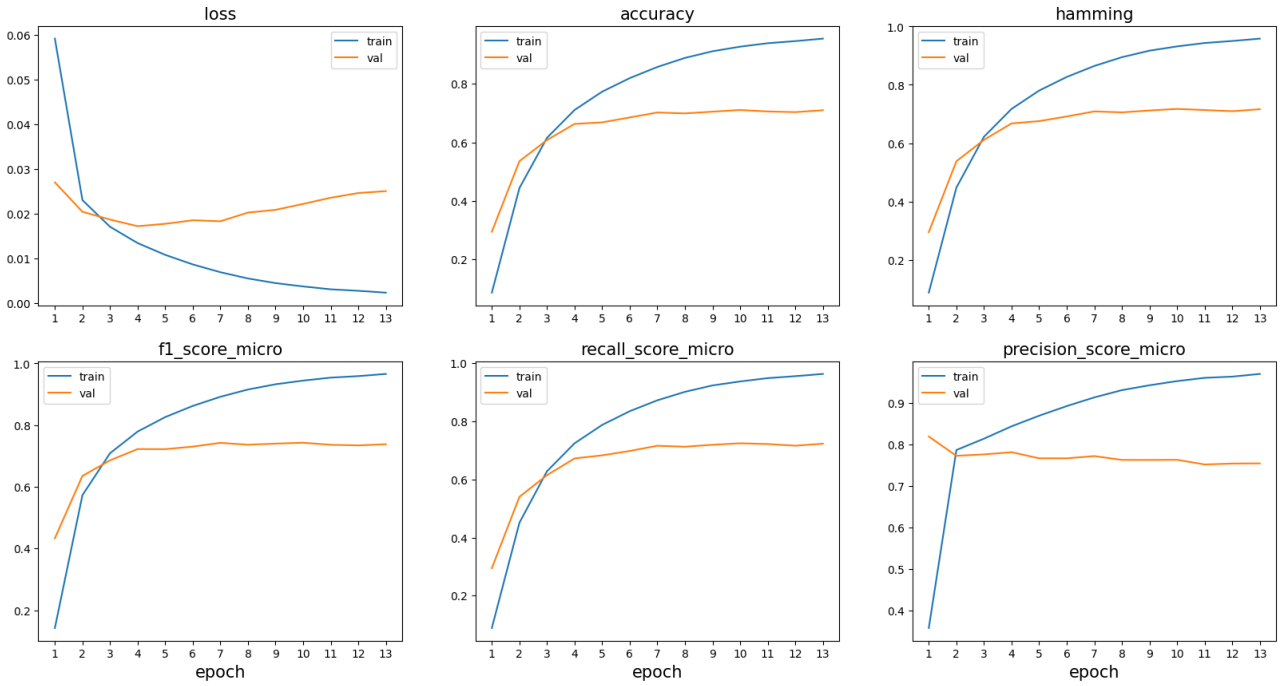

Рис. 7. Кривые обучения DistilBERT

Accuracy: 0.70976
Hamming distance: 0.71629
Identity (by Hamming distance): 0.99435

	precision	recall	f1-score	support
Соцсети	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.

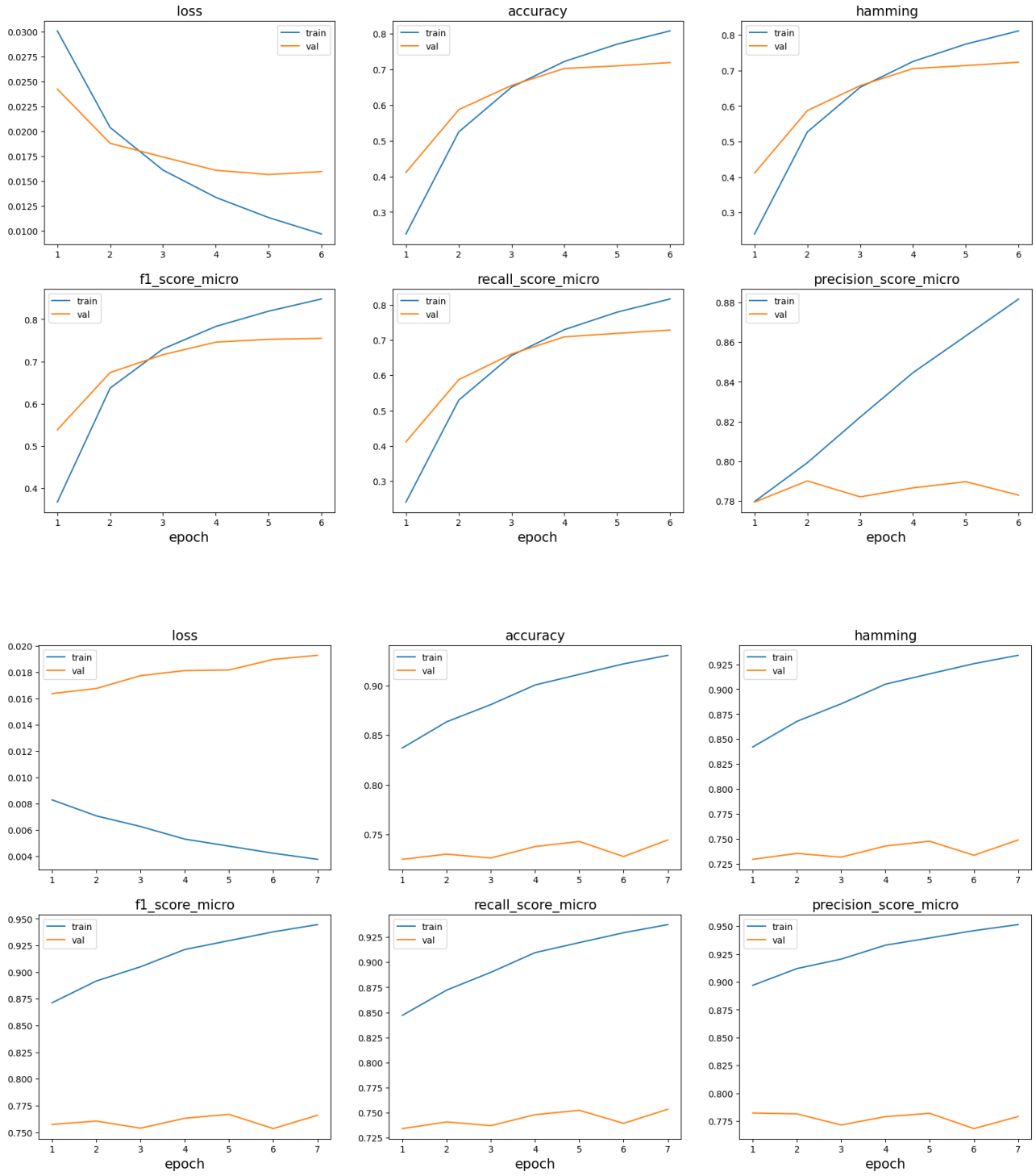

Рис. 8. Кривые обучения XLM-RoBERTa

Accuracy: 0.74312
Hamming distance: 0.74774
Identity (by Hamming distance): 0.99496

	precision	recall	f1-score	support
Соцсети	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.

	model_name	Accuracy	Hamming_distance	F1_micro	F1_macro	Recall_micro	Recall_macro	Precision_micro	Precision_macro	Train time, min.	Size, MB
0	tfidf_logreg_ngram_1_1	0.3674	0.4790	0.5238	0.5169	0.6250	0.6180	0.4508	0.4969	2	110
1	tfidf_logreg_ngram_1_2	0.3826	0.4946	0.5526	0.5327	0.6364	0.6292	0.4884	0.5150	30	2360
2	tfidf_catboost_ngram_1_1	0.4242	0.4280	0.5672	0.4639	0.4318	0.4270	0.8261	0.5590	169	661
3	ai-forever-rubert-base	0.5606	0.5606	0.5932	0.5179	0.5606	0.5543	0.6298	0.5449	365	680
4	cointegrated-rubert-tiny2	0.5038	0.5057	0.5654	0.4823	0.5076	0.5019	0.6381	0.5216	59	111
5	FacebookAI-xlm-roberta-base	0.5455	0.5474	0.5812	0.4938	0.5492	0.5431	0.6170	0.4853	778	1030
6	Geotrend-distilbert-base-ru-cased	0.5227	0.5284	0.5640	0.4931	0.5341	0.5281	0.5975	0.5059	261	208

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_ng
ram_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 Министе
рства обороны до Военной прокуратуры. Соответствующий указ в среду, 12 декабря, опубликован на официальном пор
тале правовой информации. Кроме того, еще 16 генералам присвоены очередные воинские звания. В частности, в Мин
обороны России появился один новый генерал-полковник и один полный адмирал, восемь генерал-лейтенантов и два п
риравненных к ним вице-адмирала, 10 генерал-майоров и один контр-адмирал. В МЧС появиллось два генерал-полковн
ика, четыре генерал-лейтенанта и семь генерал-майоров. В свою очередь, Росгвардия пополнилась одним генерал-по
лковником, 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 Це
нтрального военного округа стал генерал-полковником, а командующий Балтийским флотом вице-адмирал Александр Но
сатов отныне — полный адмирал. Командующий Уральским округом Росгвардии генерал-лейтенант Александр Попов полу
чил на погоны третью шитую звезду и стал генерал-полковником. В основном генеральские звания присвоены офицера
м, 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 в
нимания — красавица Мари, главная героиня сказки. Фигура главного положительного героя — Щелкунчика — установл
ена у праздничной елки возле Barvikha Hotel & Spa, украшенной бутафорскими конфетами, игрушками, бантами и кор
обками с подарками. Его сказочный антогонист — семиголовый Мышиный Король — венчает стилизованный торте праздн
ичной карусели. Карусель «оккупировала» армия Мышиного короля, охраняющая золотые орехи и бронзовые пушки. Одн
а из самых романтичных инсталляций — па-де-де, которое танцуют Фея Драже и принц Оршад: их силуэты мягко подсв
ечены. 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 откры
л традиционный рождественский базар. Фасад универмага «перевязали» красными лентами с бантами, а стены украсил
и рождественскими звездами. В атриуме ЦУМа возвышается десятиметровая новогодняя елка с золотистыми украшениям
и, 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
2. Hamming score
3. Лемматизируй это быстрее (PyMorphy2, PyMystem3 и немного магии)
4. Github. iterative-stratification
5. Multi Label Binary Classification with CatBoost
6. Hugging Face
7. Все, что нужно знать об ALBERT, RoBERTa и DistilBERT
8. Исходный датасет (800 000 строк)
9. Обработанный датасет (70889 строк)