**Слайд 1: приветствие**

Всех приветствую, меня зовут Денис Кабанов и сегодня я буду рассказывать о том, как я выполнял финальный проект по предмету “обработка естественного языка” и что у меня получилось в итоге.

**Слайд 2: основные шаги**

Перейдём к основным шагам работы, а они следующие:

Это очистка датасета, векторизация данных, обучение модели и подсчёт метрик.

**Слайд 3: очистка датасета**

Очистка являлась одним из важнейших этапов работы, без неё набрать хороший F-score не представляется возможным. Принцип моей очистки заключался в следующем:

Во-первых — определялись важные для каждой темы слова, что отличают их друг от друга.

Во-вторых — в датасете оставлялись только те записи, что включали хотя бы одно важное слово для своего класса.

При этом сами записи оставлялись почти без изменений, то есть не проводилась ни лемматизация, ни удаление стоп слов или приведение к нижнему регистру.

Стоит отметить, что сэмплы из малопредставленных классов либо удалялись, либо сливались с другими одноимёнными, классами.

**Слайд 4: очистка датасета**

Расскажу подробнее про сам принцип очистки, он состоял из пяти шагов:

Первый шаг: считалось распределение токенов по датасету в соответствии с представленной формулой, где число встреч токена в теме делилось на число записей по теме.

На втором шаге проходила первичная очистка слов, то есть оставлялись только те токены, что встречались не слишком редко и не слишком часто.

Третьим шагом было определение списка важных слов для всего датасета. Слово считалось важным, если разница максимального и среднего значения представленности токена по темам было выше порога.

**Слайд 5: очистка датасета**

После шага 3 было найдено 72 слова, что вносят основной смысловой вклад в посты пользователей. Они представлены на этом слайде, пока без разделения на важность для классов.

**Слайд 6: очистка датасета**

Четвёртым шагом был перевод частоты встречи слова с помощью Softmax в вероятность принадлежать классу. На самом деле это просто вызов операции Softmax для каждого слова.

После чего шёл заключительный пятый шаг, на котором находились важные токены для каждого класса в частности. В моём варианте просто брались первые 15 слов с наибольшей вероятностью принадлежать определённому классу.

**Слайд 7: очистка датасета**

На данном же слайде уже представлены слова, важные для каждой из тем.

**Слайд 8: очистка датасета**

После очистки датасет стал выглядеть следующим образом. В нём осталось всего лишь 20894 сэмпла, но при этом каждый из них содержал хотя бы одно важное слово. То есть была произведена очистка от шумных данных.

**Слайд 9: векторизация данных**

Перейдём к основным моментам векторизации данных.

Она проводилась с помощью WordPiece токенизатора, идущего вместе с рассматриваемой моделью с Hugging Face.

Оптимальная длина последовательности токенов определялась автоматически ещё при очистке, как квантиль уровня 0.95, и составила 330 токенов на сэмпл.

При недоборе токенов пустые места заполнялись паддингами, а при переборе обрезались с конца.

Что же до таргетов — так они кодировались с помощью One-hot encoder\_а.

Ни Upsample, ни Downsample не проводились, так как в рассмотренном подходе они хоть и улучшали предсказания на плохо представленных классах, но при этом чуть сильнее ухудшали на остальных, тем самым уменьшая Macro F-score.

Разбиение на обучающую и тестовую выборки проходило в соотношении 80 на 20.

**Слайд 10: векторизация данных**

На данном слайде приведён пример токенов, полученных после токенизации одного сэмпла. Что-то подобное получилось и с другими записями. Стоит также упомянуть, что токенизатор выдавал ещё и attention mask, но в данном случае она использовалась лишь для зануления входной последовательности на месте паддинга.

**Слайд 11: векторизация данных**

После проведение векторизации, данные на вход модели получились следующие: X — это input\_ids и attention\_mask

Y — One-hot вектора таргетов

И X и Y далее были преобразованы в torch tensor\_ы и отправлены на соответствующий девайс — GPU.

На графике ниже показано, что распределение таргетов в обучающей и тестовой выборке имеют одинаковое соотношение.

**Слайд 12: обучение модели**

Перейдём к следующей важной части проекта, а именно — к выбору модели и её обучению.

Хоть при выполнении работы и было рассмотрено несколько моделей, приоритетной оказалась именно rubert-tiny-toxicity. На это есть несколько причин. Во-первых — она основана на модели BERT, поэтому имеет все её сильные аспекты, такие как Encoder base архитектура, наличие позиционных эмбеддингов и так далее.

Во-вторых, модель была предобучена на русскоязычных текстах и посему имеет словарь с качественными эмбеддингами.

Размер же модели при этом довольно мал, всего 11.8M параметров, поэтому она спокойно умещается на локальном GPU и быстро обучается.

Ну и вишенкой на торте было то, что она выдавала хорошие метрики в задаче классификации, для которой изначально обучалась.

**Слайд 13: обучение модели**

Скажу пару слов про особенности обучения данной модели.

Оно проводилось на локальном GPU. В качестве функции потерь использовалась кросс-энтропия, а отслеживаемой метрикой был Macro F-Score на валидационной выборке. Лимит на число эпох обучения — всего 50, с автоматическим прерыванием при пяти эпохах без улучшения отслеживаемого F-score. Далее сохранялась и загружалась для оценивания только лучшая найденная модель.

**Слайд 14: подсчёт метрик**

Перейдём к итогам обучения. На данном слайде представлены три графика с отслеживаемыми метрики как на обучающем датасете, так и на валидационном. Первый график отражает изменение loss функции в зависимости от эпохи, тогда как второй и третий — accuracy и F-score соответственно. По ним можно заметить, что модели не потребовалось много времени на достижение своего оптимального состояния на валидационных данных. На это ушло всего лишь порядка пяти эпох.

**Слайд 15: подсчёт метрик**

На этом же слайде отображены более подробные предсказания модели на валидационной выборке. Как видно, F-score для каждого класса варьировался от 0.49 до 0.86 в зависимости от представленности. Тем не менее, его Macro версия оказалась равной 0.67, тогда как взвешенная по дисбалансу классов аж 0.78.

**Слайд 16: дополнительно рассмотренные подходы**

Стоит также кратко упомянуть, какие подходы были рассмотрены при решении поставленной задачи, но не возымели должного эффекта. К ним относятся:  
Использование hand-made моделей;  
Лемматизация, удаление стоп-слов, приведение к нижнему регистру;  
Использование других токенизаторов и эмбеддингов;  
Resample данных для выравнивания классов;  
Работа не с токенами, а с TF-IDF матрицей.

Все перечисленные методы и их комбинации могли как улучшать отдельные метрики для классов (precision, recall, f-score), так и ухудшать. Но общий Macro F-score у них всё равно был хуже, чем тот, что представлен сейчас.

**Слайд 17: возможности по улучшению**

Возможно ли как-нибудь ещё улучшить F-score? Ответ — да, возможно. Поэтому на данном слайде приведены несколько потенциальных подходов для улучшения качества предсказаний.

Самый лёгкий из них — это поиграть с параметрами очистки датасета, чтобы изменить важные слова для тем. Может быть их окажется больше или, наоборот, меньше. В любом случае — датасет немного видоизменится.

Следующий вариант — ручная очистка данных, чтобы быть на 100% уверенными в удалении только мусорных комментариев. Однако данный подход довольно времязатратный, поэтому следующий — это просто найти готовую или обучить самостоятельно новую модель с целью распознавания флуда.

Ну и последний вариант — взять иную модель для классификации комментариев, у которой будет другой токенизатор с эмбедднгами и большее число параметров. В надежде на то, что модели удастся обнаружить более глубокие связи между постами пользователей и их таргетами.

**Слайд 18: спасибо за внимание**

На этом всё, спасибо за внимание! Готов ответить на ваши вопросы.