Классификатор негативных комментариев

Цель проекта: Создание эффективного классификатора негативных комментариев с акцентом на сарказм и иронию.

В нашей работе мы сосредоточились на распознавании иронии и сарказма в негативных комментариях. Для этого были выбраны два подхода: использование LLM и моделей BERT/RoBERTa. LLM представляет собой современную и востребованную область, в то время как BERT/RoBERTa продемонстрировала высокие результаты в аналогичных исследованиях (Devlin et al., 2018; Liu at al., 2019; Boy et al., 2021).

В качестве данных выбрали датасет «Tweets with Sarcasm and Irony», содержащий около 85,000 твитов, классифицированных как сарказм, ирония, образное выражение («figurative») или обычное высказывание. Хотя объем данных значителен и классы разнообразны, у датасета есть свои ограничения: разметка не всегда ясна (особенно вызывает вопросы класс «figurative»). Эмоциональная окраска негативных комментариев может как усложнять, так и облегчать процесс выявления сарказма или иронии. В дальнейшем мы подробнее рассмотрим каждый из выбранных нами методов и сделаем общий вывод.

LLM

Для работы с LLM мы создали .txt файлы, так как GPT-4-turbo и DeepSeek имеют ограничения на загрузку файлов и их объём. Из-за ограничения на ввод символов, LLM получали фрагменты выборок.

Процесс работы был следующим: мы узнали, знает ли LLM о таких классах, знает ли она об иронии и сарказме. Предложили дать ей на вход тексты, которые необходимо разметить по классам, давали некоторое количество текстов, ждали ответ, давали следующее, результаты скопировали в файл.

Мы столкнулись с несколькими сложностями, которые вынудили нас отказаться от дальнейшей работы с этим способом. Для начала мы пробовали давать по 200 примеров, но, начиная с 40-ого текста, примерно, модели начинали ставить один и тот же класс. Таким образом, мы решили загружать в модели по 20 текстов. Это уже достаточно большой минус, поскольку работать с таким маленьким объемом достаточно сложно. После получения результатов в обеих LLM были проблемы с дообучением. Мы предложили каждой модели изучить примеры предложений с классами, чтобы она могла найти закономерности в дальнейшем. Однако после утверждения этого плана, модель продолжала сама выдавать классы для текстов. Структура ввода ей была объяснена, как и задание. К сожалению, даже после подтверждения задания модель не приходила к нужному результату.

Теперь посмотрим на нюансы того, что мы всё же смогли сделать при работе с LLM. GPT-4-turbo: огромный минус заключался в том, что модель начинала придумывать свои классы. Ближе к концу нам встречались такие классы, как «peace», «war», «sarcastic». В этом случае мы напоминали модели о том, какие классы есть и просили заново обработать данные, что опять же замедляло работу. DeepSeek: было намного проще работать, чем с предыдущей моделью, но порой он не все тексты брал в расчёт. Иногда вместо 20 лейблов он мог выдать 17.

Classification Report GPT EMOJI:

	precision	recall	f1-score	support
figurative	0.03	0.12	0.05	26
irony	0.68	0.49	0.57	139
regular	0.40	0.71	0.51	56
sarcasm	0.79	0.44	0.57	179
accuracy			0.47	400
macro avg	0.48	0.44	0.42	400
weighted avg	0.65	0.47	0.53	400

Classification Report DEEPSEEK EMOJI:

	precision	recall	f1-score	support
figurative	0.04	0.27	0.07	15
irony	0.66	0.49	0.56	136
regular	0.66	0.70	0.68	94
sarcasm	0.77	0.50	0.60	155
accuracy			0.53	400
macro avg	0.53	0.49	0.48	400
weighted avg	0.68	0.53	0.59	400

Поскольку времени данный подход занял очень много, а к дообучению мы так и не смогли дойти, то решено было остановиться на том, что мы сделали. То, как мы формировали текстовые форматы можно посмотреть здесь: Work with LLM.ipynb

ToXCL и Fuzzy rough nearest neighbour methods

Мы решили ознакомиться с другими исследованиями, чтобы понять, в каком направлении можем двигаться. В первую очередь, изучив статью (N. Hoang et al. 2024), мы попытались разобраться с фреймворком ToXCL. Наша цель заключалась в том, чтобы запустить его и выяснить, как функционирует такая структура. Однако мы столкнулись с трудностями из-за отсутствия предобученных моделей, которые требовались в нескольких экземплярах и не были доступны в открытом доступе. Запросить их тоже не оказалось возможным. Кроме того, формат этого фреймворка не совсем подходил нам, так как он больше ориентирован на выявление социальных предубеждений и связан с генерацией текста, а не с классификацией. Авторы достигли итоговой точности 81.53 (Acc) и 78.19 (F1) с использованием модели T5.

Еще одним методом, который мы рассмотрели, был подход, основанный на теории нечетких приближений для классификации (O. Kaminska et al. 2023). Суть этого метода заключается в том, что для каждого тестового примера вычисляются степени принадлежности к нижним и верхним приближениям классов. К сожалению, ссылка на репозиторий с кодом оказалась недействительной, и нам не удалось воспроизвести результаты. На данных из соревнований авторы получили результат по иронии (сарказм не рассматривался отдельно) на уровне 0.9365 (F1). Это довольно высокий показатель, поэтому мы решили не углубляться в этот метод и попробовать разработать что-то свое. Тем не менее, мы будем использовать этот результат в качестве бейзлайна и постараемся превзойти его нашим подходом.

BERT and RoBERTa

Проанализировав работу Boy et al. (2021), мы поняли, что нужно пробовать работать с BERT и RoBERTa. Для начала мы взяли модели BertForSequenceClassification и RoBERTaForSequenceClassification. Эти модели показали хорошие результаты в плане общей точности, но мы столкнулись с проблемой баланса между precision и recall для классов "сарказм" и "ирония". Модель была очень осторожна и значение recall было невысоким, хотя precision был равен 1.

После получения фидбека на промежуточной защите и анализа результатов, мы решили перейти к использованию модели Sentence-RoBERTa. Это решение было обусловлено несколькими факторами:

1. Sentence-RoBERTa демонстрирует более сбалансированные показатели точности и полноты для классов "сарказм" и "ирония", что важно для нашей задачи.
2. Sentence-RoBERTa позволяет лучше обобщать на новые данные, что особенно полезно при работе с разнообразными комментариями.
3. Хотя общая точность Sentence-RoBERTa немного ниже, её способность более сбалансированно распознавать сарказм и иронию делает её более подходящей для нашего проекта.

Таким образом, Sentence-RoBERTa стала наиболее подходящей базовой моделью для нашего проекта.

Затем мы провели серию экспериментов, чтобы определить оптимальную архитектуру модели на основе Sentence-RoBERTa. Мы начали с базовой конфигурации, включающей линейный слой с дропаутом и нормализацией, и оценили её производительность. Хотя модель показала высокую точность для классов наших целевых иронии и сарказма, результаты для класса «figurative» оказались крайне низкими.

В попытке улучшить результаты, мы добавили механизм attention, чтобы модель могла лучше фокусироваться на важных частях текста. Однако, несмотря на небольшое улучшение для класса

«regular», результаты для класса «figurative» остались неудовлетворительными. Далее мы попробовали использовать multihead attention, чтобы улучшить обработку текста, но это привело к еще большему ухудшению результатов для класса «figurative» .

В итоге, несмотря на изменения в архитектуре модели, результаты для класса «figurative» остались без улучшений. Мы осознали, что исходили из ошибочной гипотезы, полагая, что архитектура модели является ключевым фактором успеха. Поэтому мы решили сосредоточить больше усилий на анализе данных.

Мы поняли, что проблема может быть связана с дисбалансом классов в наших данных. Для решения этой проблемы мы применили взвешивание классов, чтобы модель уделяла больше внимания меньшему классу, а также создали эмбеддинги текстов с помощью SentenceTransformer. В качестве альтернативного подхода мы использовали undersampling для выравнивания количества примеров в каждом классе. После обучения модели с использованием этих методов мы заметили улучшение recall для класса «figurative», однако precision остался на низком уровне. Это свидетельствует о том, что модель по-прежнему сталкивается с трудностями в правильной классификации. В результате мы поняли, что необходимо улучшить процесс создания эмбеддингов.

Мы занялись Fine-tuning-ом. Нашей задачей было работать с классификационным слоем, поэтому параметры предобученной модели зафиксировали. В итоге внесли минимальное изменение, добавив линейный слой, это не сильно повлияло на наши ресурсы, но чуть-чуть улучшило результаты. На первых тестовых запусках он помог нам сбалансировать распознавание иронии и сарказма. Тем не менее, проблема с классом «figurative» оставалась нерешенной. Мы не могли понять правила разметки для этого класса и, просматривая данные, не обнаружили каких-либо его характерных особенностей. К сожалению, автор датасета не предоставил никакой информации о разметке: была ли она ручной или автоматической, и каким образом определялись классы. Модель также испытывала трудности, что отражалось на очень низких значениях. Вот несколько примеров из класса «figurative»:

(1) My life today 😊👌 #sarcasm

(2) Women are so.....simple..... #females #complex #irony #follow #wtf http://t.co/iDrvoOIJJW

Складывается впечатление, что разметка данного класса слишком субъективна или даже произвольна. В результате, этот класс мешал модели, и нам не удавалось добиться хороших результатов в разметке других классов. Мы решили отказаться от работы с ним, так как нашей изначальной целью было различение иронии и сарказма. Мы сосредоточились на целевых классах: ирония, сарказм и обычное выражение.

Sentence RoBERTa – финальный продукт

В нашем проекте мы разработали модель для классификации твитов на три категории: «regular», «irony» и «sarcasm». Мы начали с подготовки данных. Для борьбы с дисбалансом классов мы вычислили веса классов, которые затем использовали в функции потерь, чтобы модель уделяла больше внимания меньшим классам.

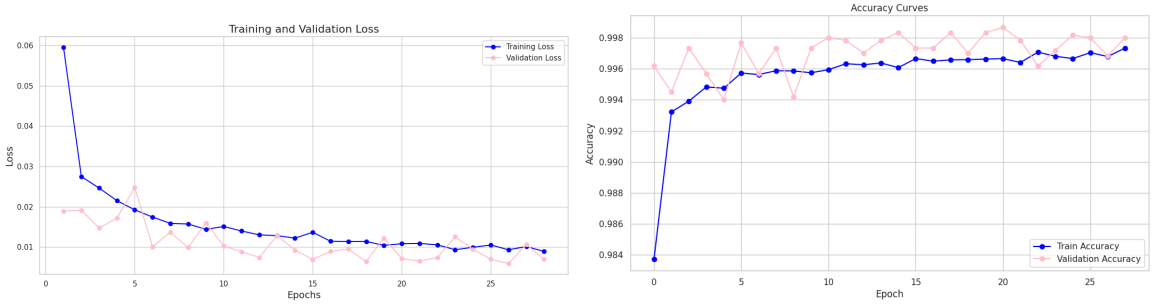
Мы пересчитали эмбеддинги для нашего укороченного датасета, используя предобученную на наших данных модель SentenceTransformer, что позволило нам получить более качественные представления текстов. Эти эмбеддинги затем подаются на вход классификатора SBERT, состоящего из нескольких слоев: Batch Normalization для нормализации входных данных, Dropout для предотвращения переобучения и линейный слой для преобразования эмбеддингов в логиты для каждого из трех классов.

Мы также используем взвешенную функцию потерь, которая учитывает дисбаланс классов, позволяя модели уделять больше внимания меньшим классам и улучшая общую производительность классификации. Модель выводит предсказанные классы для каждого твита на основе логитов, полученных от линейного слоя классификатора.

Процесс подбора гиперпараметров включал использование библиотеки Optuna для автоматического поиска оптимальных значений таких параметров, как learning_rate, batch_size и dropout_rate, причем для learning_rate использовался метод логарифмического равномерного распределения, а для dropout_rate применялось равномерное распределение в диапазоне от 0.0 до 0.7, что позволило нам более точно настроить модель и улучшить её производительность. В результате были выбраны лучшие гиперпараметры, которые затем использовались для окончательного обучения модели:

{'learning_rate': 0.003713, 'batch_size': 32, 'dropout_rate': 0.01023}

Обучение модели проводилось в течение 100 эпох с использованием раннего прекращения (early stopping), чтобы избежать переобучения. Мы отслеживали потери и точность на каждой эпохе, что позволило нам анализировать производительность модели и своевременно останавливать обучение, если улучшений не наблюдалось. В результате алгоритм сработал на 28-й эпохе, что действительно помогло предотвратить переобучение и сохранить оптимальные характеристики модели.



Визуализация потерь и точности на тренировочном и валидационном наборах показала, что модель хорошо обобщает данные и не переобучается. Анализ ошибок на валидационном и тестовом наборах выявил, что модель допускает редкие ошибки в классификации некоторых твитов, что является нормальным явлением при работе с такими сложными категориями, как сарказм и ирония. Мы решили проанализировать ошибки, сделали это здесь: misclassified_samples_test . Общий тренд в анализе ошибок на тестовых данных показывает, что многие ошибки связаны с субъективностью разметки и интерпретацией текста. Некоторые твиты, размеченные как ироничные, на самом деле не содержат иронии, что может быть связано с недостаточной ясностью или неправильной интерпретацией текста. К сожалению, о процедуре разметки данного датасета ничего неизвестно, так что подобные случаи субъективности разметки вполне ожидаемы.

На тестовом наборе данных модель также показала высокие результаты, что подтверждает её эффективность в классификации твитов на три категории: «regular», «irony» и «sarcasm».

```
Classification Report:
              precision    recall  f1-score   support

     regular       0.99      1.00      0.99      1859
       irony       1.00      0.99      0.99      2111
     sarcasm       1.00      1.00      1.00      2105

    accuracy                           1.00      6075
   macro avg       1.00      1.00      1.00      6075
weighted avg       1.00      1.00      1.00      6075
```

Таким образом, наш подход с использованием весов классов, пересчёта эмбеддингов и тщательного подбора гиперпараметров позволил создать эффективный классификатор негативных комментариев, способный распознавать сарказм и иронию.

Код со всеми этапами визуализации и анализом представлен здесь: Sentence-RoBERTa (final).ipynb
Все материалы собраны в этой папке: Классификатор негативных комментариев
Таймлайн проекта можно увидеть здесь: Классификатор негативных комментариев

Литература

Boy, S., Ruiter, D., & Klakow, D. (2021). Emoji-based transfer learning for sentiment tasks. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop* (pp. 103–110). Association for Computational Linguistics.

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv*. Cornell University.

Hoang, N., Do, X., Do, D., Vu, D., & Luu, A. (2024). ToXCL: A unified framework for toxic speech detection and explanation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (Vol. 1, pp. 6460–6472). Association for Computational Linguistics.

Kaminska, O., et al. (2023). Fuzzy rough nearest neighbour methods for detecting emotions, hate speech and irony. *Information Sciences, 625*, 521–535.

Lima, L., Pagano, A., & Silva, A. (2024). Toxic content detection in online social networks: A new dataset from Brazilian Reddit communities. In *Proceedings of the 16th International Conference on Computational Processing of Portuguese - Vol. 1* (pp. 472–482). Association for Computational Linguistics.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. *arXiv*. Cornell University.

Mishra, S., & Chatterjee, P. (2024). Exploring ChatGPT for toxicity detection in GitHub. In *IEEE/ACM 46th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)* (pp. 6-10). IEEE/ACM.

Oliveira, A., Cecote, T., Alvarenga, J., Freitas, V., & Luz, E. (2024). Toxic speech detection in Portuguese: A comparative study of large language models. In *Proceedings of the 16th International Conference on Computational Processing of Portuguese - Vol. 1* (pp. 108–116). Association for Computational Linguistics.