Проект ML

Главное:

Мы решаем задачу NLP по классификации.

Аннотация

Мы работаем с датасетом "Natural Language Processing with Disaster Tweets"на Kaggle. Наша задача проанализировать размеченный список публикаций с Twitter и обучить модель классифицировать, несёт ли публикация информацию о катастрофе. Датасет полностью на английском. Состоит из 4 столбцов:

- `id` - уникальный номер для каждой публикации
- `text` - текст публикации
- `location` - откуда опубликовано (может быть пусто)
- `keyword` - ключевое слово (может быть пусто)
- `target` - только в тренировочном датасете, обозначает, несёт ли публикация сообщения о катастрофе (1) или нет (0)

Введение

Для решения задачи мы использовали Google Collab, где обучили модель трансформер из huggingface с замороженным бэкбоном.

Сначала мы анализировали предложенный датасет, потом очистили его с помощью регулярных выражений, так, например, были удалены символы неанглийского алфавита, обращения к другим пользователям и ссылки (они, очевидно, не несут полезной информации). Колонка "местоположение" и "номер" были удалены, а колонка "ключевое слово" была объединена с колонкой "текст", ключевые слова добавлялись в конце текста публикации, если были соответственно.

Далее создали копию тренировочного датасета и первую токенизировали с помощью предобученного токенизатора модели *roberta-large-mnli*, а вторую - с *bert-base-uncased* . Теперь, подготовив набор данных, мы обучили модели и сравнинили их.

Для обучения были выбраны модели "transformer", состоящие из нескольких кодировщиков. Они, в отличие от рекуррентных нейронный сетей обладают механизмом внимания, не забывают контекст с течением времени.

Обозначения, терминология

Датасет - набор данных

Бэкбон (backbone) - основная сеть, служащая для извлечения признаков из поступающего на вход изображения. Предобученная модель

Замороженный бэкбон - Бэкбон замораживается и не обновялется в процессе обучения

NLP (Natural Language Processing) - Обработка естественного языка

Трансформер (Transformer) - архитектура глубоких нейронных сетей, представленная впервые в 2017 году в статье Attention is all you need

Google Collab - бесплатная облачная платформа для создания и выполнения кода на Python с возможностью использования центральных и графических процессоров

Huggingface -американская компания, разрабатывающая инструменты для создания приложений с использованием машинного обучения

Токенизация - это процесс разделения текста на более мелкие части, такие как слова или предложения. Этот процесс позволяет преобразовать непрерывный текст в дискретные элементы, с которыми можно работать отдельно.

ф1 метрика (f1_score) - в отличии от точности учитывает ещё и дисбаланс классов гармоническое среднее между точностью (Precision) и полнотой (Recall) в машинном обучении

$$F1_{score} = 2 * \frac{(Precision * Recall)}{(Precision + Recall)}$$

Precision - метрика, которая показывает долю объектов, классифицированных как "верные" и при этом действительно являющихся положительными

Recall - метрика, которая показывает, какую долю объектов положительного класса из всех объектов положительного класса нашёл алгоритм

Батч (Batch) - подмножество тренировочных данных, используемое в одной итерации обновления параметров модели

Эпохи - полная итерация по набору данных в процессе обучения модели

Обзор литературы

Архитектура Bert - статья для понимания архитектуры Bert RoBERTa - статья на английском, в ней рассказывается об отличиях от модели Bert

Работа

В ходе работы было проведено несколько экспериментов

1 эксперимент

Самым удачным оказалось обучение модели *bert-base-uncased* с замороженным бэкбон.

Она обучалась 15 эпох 49 минут и показала хороший результат в 82 % точности и 81 % ф1 метрики. Именно результаты работы этой модели я использовал на Kaggle, где получил $F1_{score} = 0.8112$ (81 %) и занял в таблице лидеров 339 место.

[375/375 49:31, Epoch 15/15]

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.545600	0.808743	0.784370	0.800995	0.784370	0.771676
2	1.119500	1.223617	0.730825	0.790264	0.730825	0.693374
3	0.617000	0.772295	0.696093	0.758237	0.696093	0.695451
4	0.949600	1.159930	0.703329	0.774946	0.703329	0.652064
5	0.528400	0.499053	0.817656	0.816602	0.817656	0.816095
6	0.564600	0.504178	0.811867	0.810901	0.811867	0.809770
7	0.969100	1.426244	0.568741	0.751769	0.568741	0.530959
8	1.232100	0.608037	0.768452	0.783738	0.768452	0.770703
9	0.706100	0.508716	0.807525	0.809095	0.807525	0.808103
10	0.726400	0.452463	0.810420	0.811646	0.810420	0.810893
11	0.504000	0.516380	0.765557	0.783283	0.765557	0.767877
12	0.500700	0.457056	0.811867	0.812102	0.811867	0.811977
13	0.449800	0.450179	0.810420	0.812323	0.810420	0.811081
14	0.533700	0.428337	0.819103	0.818540	0.819103	0.816840
15	0.463500	0.431114	0.820550	0.823601	0.820550	0.816144

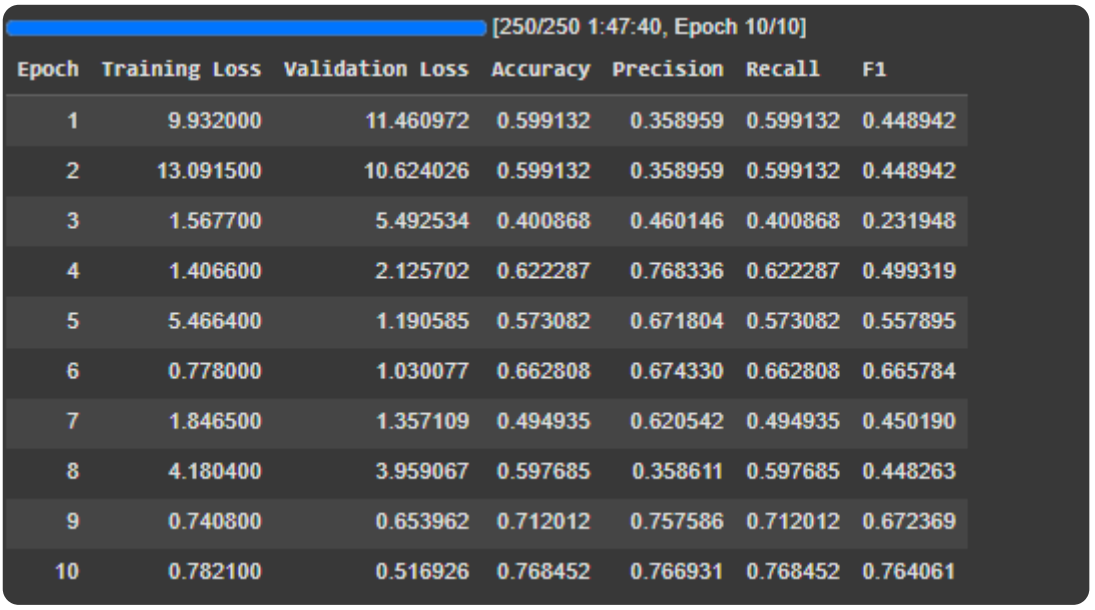
2 эксперимент

Далее я разморозил бэкбон и попытался обучить модель с теми же параметрами.

Но модель стала занимать слишком много памяти графического процессора. Для её обучения пришлось уменьшать размер батча с 256 до 8. Качество модели значительно упало, а время сильно увеличилось. В итоге данная модель не вошла в итоговую работу, но заслуживает упоминания.

3 эксперимент

Но почему я выбрал именно *bert-base-uncased*? Потому что это относительно простая модель, время обучения которой должно быть небольшим. Первоначально планировалось обучить только модель *roberta-large-mnli* с замороженным и размороженным бэкбоном. Roberta превосходит стандартный bert как минимум по тому, что была обучена на количестве данных в 10 раз больших, чем bert. С неё и начались эксперименты. Обучение Roberta с размороженным бэкбоном приведено в первоначальной версии прикреплённого ноутбука. Эта модель обучалась очень долго. Я пробовал её обучил успешно два раза. На 10 эпох и на 20 эпох. 10 эпох:


[250/250 1:47:40, Epoch 10/10]

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	9.932000	11.460972	0.599132	0.358959	0.599132	0.448942
2	13.091500	10.624026	0.599132	0.358959	0.599132	0.448942
3	1.567700	5.492534	0.400868	0.460146	0.400868	0.231948
4	1.406600	2.125702	0.622287	0.768336	0.622287	0.499319
5	5.466400	1.190585	0.573082	0.671804	0.573082	0.557895
6	0.778000	1.030077	0.662808	0.674330	0.662808	0.665784
7	1.846500	1.357109	0.494935	0.620542	0.494935	0.450190
8	4.180400	3.959067	0.597685	0.358611	0.597685	0.448263
9	0.740800	0.653962	0.712012	0.757586	0.712012	0.672369
10	0.782100	0.516926	0.768452	0.766931	0.768452	0.764061

20 эпох:

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	10.305300	11.951322	0.599132	0.358959	0.599132	0.448942
2	12.592400	9.992604	0.597685	0.358611	0.597685	0.448263
3	1.116600	4.659231	0.599132	0.358959	0.599132	0.448942
4	3.453400	4.669896	0.599132	0.358959	0.599132	0.448942
5	4.580100	2.594554	0.567294	0.534905	0.567294	0.532632
6	1.395000	1.193070	0.638205	0.774423	0.638205	0.531393
7	1.090600	1.218599	0.622287	0.683664	0.622287	0.511768
8	0.947700	0.778547	0.645441	0.687812	0.645441	0.647033
9	5.324200	2.181677	0.461650	0.634588	0.461650	0.377971
10	4.149100	1.446784	0.617945	0.691145	0.617945	0.498954
11	3.923300	4.556591	0.599132	0.358959	0.599132	0.448942
12	2.251700	1.343438	0.636758	0.747581	0.636758	0.532364
13	0.718900	0.570683	0.764110	0.762805	0.764110	0.758995
14	3.109300	1.528565	0.664255	0.725856	0.664255	0.594413
15	0.873000	0.760654	0.623734	0.696693	0.623734	0.619303
16	0.930400	0.663370	0.735166	0.770106	0.735166	0.706645
17	0.885000	0.565336	0.690304	0.728593	0.690304	0.692339
18	0.784900	0.619988	0.707670	0.764794	0.707670	0.662498
19	0.734900	0.498731	0.756874	0.754939	0.756874	0.755378
20	0.848300	0.528310	0.752533	0.754402	0.752533	0.753276

Сложности также были вызваны с нестабильным интернет подключением, из-за которого Google Collab постоянно обновлял локальную среду, стирая все наработки

Выводы

Мы провели серию экспериментов, в ходе которых обучали разные модели и сравнивали их точность. Самой точной моделью оказалась *bert-base-uncased* с замороженным бэкбоном, которая в сравнении с другими рассмотренным показала наименьшее время работы и лучшее качество с первых эпох. *Roberta-large-mnli* по нашим предположениям должна была показать большее качество в связи с тем, что обучена она лучше, но на практике она не смогла превзойти *bert-base-uncased*. Возможно, для её работы нужно было увеличить количество эпох и размер батча, но у нас были ограничены ресурсы, поэтому мы не смогли это проверить.

Заключение

С помощью модели *bert-base-uncased* мы смогли добиться хорошего качества в 81 % Мы убедились в том, что не всегда модели, обученные лучше показывают качество

лучше.