Tinkoff Lab. Тестовое задание NLP

Куимов Михаил

kuimov.ms@phystech.edu

29 марта 2022 г.

1 Описание

В данном задании требовалось

1. Взять задачи SST-2, RTE и CoLA из GLUE [1] и обучите на них классификатор.

2. Проанализировать полученные результаты и сравнить используемые методы.

3. Сопоставить полученные результаты с результатами, которые публикуют другие люди.

2 Обзор

GLUE (General Language Understanding Evaluation) бенчмарк - набор NLU задач, включающий question answering, sentiment analysis и textual entailment, и платформу для тестирования и сравнения моделей. Рассмотрим задачи, которые предлагается решить.

- **CoLA** - задача Лингвистической Приемлемости (Linguistic Acceptability). Размеченный датасет состоит из английских предложений и лейблов, которые указывают на то, является ли предложение грамматически правильным. Для оценки качества предсказаний используется корреляция Мэтью [2]. Датасет для тренировки имеет 8.5 тысяч образцов. Для тестирования и валидации - 1 тысяча образцов.

- **SST-2** - задача Анализа Настроения (Sentiment Analysis). Датасет состоит из обзоров на фильмы, размеченных, являются ли они позитивными или негативными. Для оценки качества предсказаний используется Accuracy. Датасет для тренировки имеет 67 тысяч образцов. Для тестирования - 1.8 тысяча образцов. Для валидации - 900.

- **RTE** - задача определения Текстового Следствия (Textual Entailment). Датасет состоит из пар предложений, размеченных бинарными лейблами, которые указывают, является ли одно следствием другого. Для оценки качества предсказаний используется Accuracy. Датасет для тренировки имеет 2.5 тысяч образцов. Для тестирования - 3 тысячи образцов. Для валидации - 300 образцов.

3 Методология

В задании сказано обучить классификатор. Дополнительно прочитав статью авторов GLUE [1], я подумал, что нужно обучить мультизадачный классификатор на трех задачах[1]. Также посмотрев лидерборд[2], я выяснил, что лучшие методы основаны на предобученных енкодерах трансформеров. Поэтому я решил протестировать мультизадачную модель, основанную на файнтюнинге (fine-tune) пред- обученных моделей трансформеров и сравнить ее с монозадачной моделью схожей архитектуры. В качестве предобученной модели я решил взять BERT [3] с Hugging Face[3]. Я тестировал и другие модели, эта дала лучшие результаты.

3.1 Мультизадачная модель с одной головой классификации

Обозначим данный метод SingleHeadMultitask. Общая схема подхода показана на изображении 1. Метод файнтюнинга я взял из оригинальной статьи [1]. Мы подаем в енкодер токенизированное предложение. Последовательность токенов имеет в начале специальный [CLS] токен.

[1] Я спросил у поддержки, правильно ли я понял. Она мне ответила только через полтора дня, что нужно обучить три классификатора

[2] https://gluebenchmark.com/leaderboard

[3] https://huggingface.co/bert-base-uncased

Выходной ембединг этого токена содержит в себе всю информацию о предложении. Поэтому на выходе мы берем ембединг [CLS] токена и подаем его на вход линейного слоя[4] с нелинейностью sigmoid. Взяв argmax от выходных вероятностей классов, мы получаем метку класса. Чтобы применить данную архитектуру к случаю мультизадачной модели, были сделаны следующие модификации:

1. Для задачи RTE пара входных предложений была соединена в одно следующим образом: $sentence_1 + "[SEP]" + sentence_2$. [SEP] - специальный токен, разделяющий предложения.

2. Для каждой задачи (CoLA, SST-2, RTE) к предложениям были добавлены теги задачи[5] в формате: $tag : sentence$. Это сделано, чтобы модель отличала задачи друг от друга.

3. Был использован мультизадачный Dataloader. Он выдает батч образцов из одного из трех рассматриваемых датасетов. При этом семплы из разных датасетов не перемешиваются.

4. Значения функции ошибки для каждой задачи были нормированы размером соответствующего датасета, чтобы слагаемые были одинакового порядка.

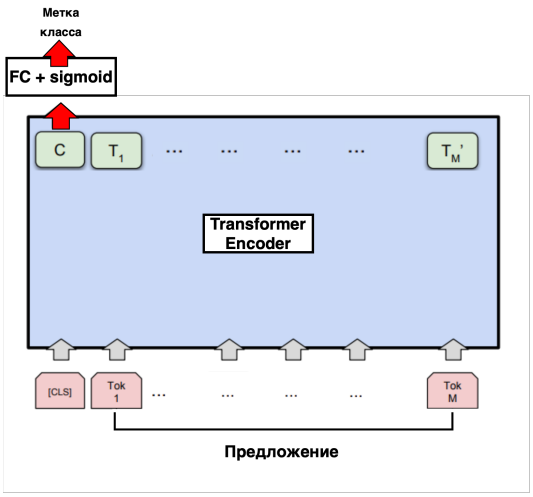

Рис. 1: SingleHeadMultitask архитектура

[4]Линейный слой имеет размерность $emb\_len \times 2$, где emb_len - это размерность hidden layer

[5]Теги: «acceptability», «sentiment», «nli»

3.2 Мультизадачная модель с тремя головами классификации

Обозначим данный метод MultiHeadMultitask. Общая схема подхода показана на изображении 2. Данный метод отличается от модели с одной головой классификации только в двух местах:

1. Теги задачи к предложениям не добавляются.

2. Модель имеет общий для всех задач енкодер. Однако, для каждой задачи есть своя классификационная голова, которые имеют одинаковую структуру, описанную в предыдущей подсекции.

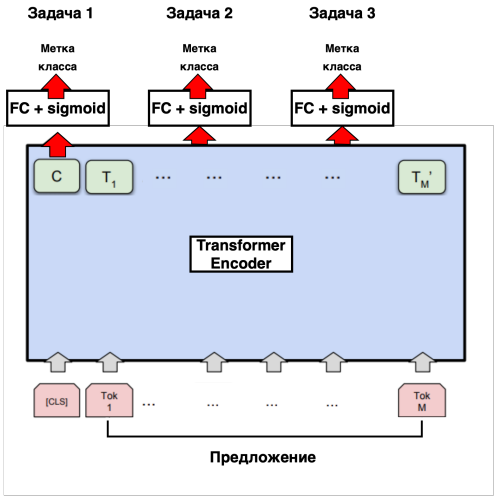

Рис. 2: MultiHeadMultitask архитектура

3.3 Монозадачная модель

Обозначим данный метод Singletask. Данный подход имеет аналогичную модели SingleHeadMultitask архитектуру. Здесь мы подаем данные из одного датасета и обучаем модель для каждой задачи отдельно.

3.4 Дополнительные улучшения

Все описанные выше модели переобучались на 2−3 эпохе обучения. Это стандартная ситуация для таких больших моделей, как BERT. Поэтому в попытке улучшить результаты, я решил сделать аугментацию (augmentation) для задач CoLA и RTE. Задача SST-2 итак имеет много данных для обучения. Чтобы добавить больше данных, были предприняты следующие шаги для каждого датасета:

1. Было выбрано 25% случайных предложений.

2. Каждое предложение было переведено на один из следующих языков: русский, немецкий, французский, испанский, китайский. Язык для каждого предложения выбирался случайно.

3. Предложение было переведено обратно на английский.

Таким образом я увеличил размер датасетов новыми семплами. Для задачи CoLA были получены только правильные грамматически новые предложения, так как переводчик при переводе создает грамматически правильные предложения. Для перевода был испольщован Google переводчик из библиотеки deep_translator[6].

3.5 Конфигурация модели

Для написания и обучения моделей был использован Pytorch [4]. Во время обучения веса пред-обученных моделей не были заморожены. Все модели обучались с использованием Бинарной Кросс-Энтропии в качестве функции ошибки, оптимизатора AdamW [5] с learning rate-ом $2 \cdot 10^{-5}$ на протяжении 10 эпох. Веса модели сохранялись при увеличении среднего значения метрик для трех рассматриваемых задач. При тестировании использовались веса модели с лучшим показателем средней метрики. Размер батча на тренировки брался равным 16, а на валидации равным 8. Также для избежании раннего переобучения использовался линейный learning rate scheduler, или warm-up по-другому.

4 Результаты

Результаты работы всех моделей представлены в таблице 1. Также прилагаю сравнение с результатами лучшего по метрикам бейзлайна из статьи про GLUE [1]. Результаты всех подходов авторов бенчмарка на валидационном датасете можно найти в аппендиксе их статьи. Я провожу именно такое сравнение, потому что для корректного сравнения на тестовом датасете, мне бы нужно было обучить модели для всех GLUE задач, чтобы загрузить их в тестирующую систему. Каждая модель тренировалась и была протестирована 3 раза, чтобы исключить вариант случайности. Так как аугментация для SST-2 не делалась, то в таблице представлен результат однозадачной модели без аугментации. Мультизадачных модели были протестированы отдельно на каждом из датасетов.

[6] https://deep-translator.readthedocs.io/en/latest/

Из таблицы 1 видно, что лучший результат был достигнут монозадачной моделью с аугментацией. Аугментация дала прирост в 4% для задачи RTE. Однако, качество для CoLA ухудшилось. Я думаю, это связано с тем, что при добавлении к тренировочной выборке образцов одного класса, классы стали менее сбалансированными. Схожее поведение наблюдается и для SingleHeadMultitask модели. Для мультизадачных моделей аугментация не дала прироста в качестве классификации. Это может быть связано с тем, что размер добавленных данных сильно меньше датасета SST-2. Поэтому модель не чувствует разницы. Несмотря на то, что монозадачные модели показали лучшие результаты, мультизадачные дают схожие результаты, при этом являясь универсальными для этих 3 задач. Также одноголовая и многоголовая модели показали очень похожие результаты. Многоголовая немного лучше. Для всех задач было достигнуто улучшение по сравнению с бейзлайнами. Самое большое улучшение было достигнуто для задачи CoLA, в 23%.

5 Обсуждение

Для улучшения результатов можно протестировать следующие идеи:

- Увеличить размер добавленных аугментацией данных.
- Не делать аугментацию на CoLA, или же добавить семплы класса неправильных грамматически предложений путем перемешивания слов в предложении.
- Сделать эксперименты на других пред-обученных моделях.

6 Заключение

Лучшие результаты были показаны монозадачной моделью с аугментацией. Но мультизадачные методы показали схожие результаты и имеют место для улучшения.

Список литературы

[1] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, “Glue: A multi-task benchmark and analysis platform for natural language understanding,” 2018. [Online]. Available: https://arxiv.org/abs/1804.07461

Model	CoLA	SST-2	RTE	Average
Singletask	**0.598 ± 0.010**	**0.927 ± 0.002**	0.616 ± 0.012	0.714 ± 0.008
Singletask augmented	0.572 ± 0.009	**0.927 ± 0.002**	**0.651 ± 0.007**	**0.717 ± 0.012**
SingleHeadMultitask	0.583 ± 0.015	0.913 ± 0.002	0.590 ± 0.007	0.695 ± 0.012
SingleHeadMultitask augmented	0.571 ± 0.008	0.917 ± 0.003	0.590 ± 0.027	0.693 ± 0.013
MultiHeadMultitask	0.557 ± 0.002	0.922 ± 0.001	0.613 ± 0.009	0.697 ± 0.013
MultiHeadMultitask augmented	0.560 ± 0.028	0.921 ± 0.001	0.615 ± 0.005	0.695 ± 0.013
Baseline	0.367	0.911	0.614	0.631 ± 0.025

Таблица 1: Качество работы моделей и сравнение с бейзлайном на валидационных данных. Для SST-2 и RTE указана accuracy, для CoLA - Mattews correlation. Average - среднее значение метрик всех 3 задач.

[2] B. Matthews, "Comparison of the predicted and observed secondary structure of t4 phage lysozyme," *Biochimica et Biophysica Acta (BBA) - Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0005279575901099

[3] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[4] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[5] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980