Named Entity Oriented Sentiment Analysis (RuSentNE-2023) - Study of RuBERT applications in faster models

Anonymous DIALOGUE submission

Abstract

The author of this article tries to study the usage of RuBERT in Named Entity Oriented Sentiment Analysis Task (RuSentNE-2023) in lighter (and therefore faster) models with lower quality of predictions. The author tries different approaches and compares the results. Best F1-score by author is 27.73.

Keywords: ML, Deep Learning, BERT, RuBert, Sentiment Analysis

DOI: none (yet)

Изучение применения RuBERT в задаче RuSentNE-2023 для более быстрых и простых решений

Спицын Николай

MIPT

spitsyn.na@phystech.edu

Аннотация

Автор этой статьи изучает использование RuBERT в задаче анализа настроений именованных объектов (RuSentNE-2023) в более простых (и, следовательно, более быстрых) моделях с более низким качеством прогнозов. Автор пробует разные подходы и сравнивает результаты. Лучший достигнутый результат - 27.73.

Ключевые слова: анализ настроений, классификация настроений текста, BERT, ruBERT, ML, Python.

1 Вступление

1.1 Постановка задачи

RuSentNE-2023 - первый конкурс по целевому анализу настроений к названным объектам в новостных текстах на русском языке. Именованные сущности классифицировались по трем классам тональности: положительной, отрицательной или нейтральной в пределах одного предложения.

Особенности новостных текстов заключаются в следующем:

- Новостные тексты содержат множество именованных сущностей с нейтральным настроением, что означает преобладание нейтрального класса;
- С другой стороны, некоторые предложения полны именованных объектов с разными настроениями, что затрудняет определение настроения для конкретного именованного объекта.

1.2 Описание датасета

Датасет представляет собой список предложений из новостных текстов СМИ. Каждое предложение аннотируется:

- *entity* — объект анализа настроений
- *entity_tag* — тег данного объекта (PERSON, ORGANIZATION, PROFESSION, COUNTRY, NATIONALITY)
- *entity_pos_start_rel* — индекс начального символа данной сущности
- *entity_pos_end_rel* — индекс следующего символа после последнего из данной сущности
- *label* — метка тональности

Каждая сущность имеет трехуровневую метку. Используются следующие классы (*label*):

- Отрицательный (-1)
- Нейтральный (0)
- Положительный (1)

При изучении датасета выяснилось, что большая часть текстов имеет нейтральный тон (4774 из 6637, порядка 72%. Ожидаемо, второе место занял негативный тон - 1007 записей). Из этой несбалансированности становится очевидно применение *F1-метрики*, так как

2 Related work

Схожую работу провели в 2020 году и лучших результатов добились при использовании SVM + Word2VEC (skipgram) embedding (Bade Shrestha and Bal, 2020). Авторы не пробовали применять другие embedding'и, такие как BERT. В России так же есть схожие публикации - авторы применяли BERT для анализа тональности длинных русских новостных текстов (Kotelnikova and Kropanev, 2021).

Аналогично уже были проведены исследования по оценке качества различных моделей BERT, приведённых на huggingface.co: (Kosykh, 2022)

Кроме того, похожий анализ был уже проведён авторами соревнования: (Golubev and Lukashevich, 2021)

3 Model setup

Первая использованная модель использовала Bert исключительно для получения embedding'ов, после чего использовались catboost и LogisticRegression для получения финального ответа (label'а). Модель имела следующий вид:

```
BertModel(
  (embeddings): BertEmbeddings(
    (word_embeddings): Embedding(119547, 768, padding_idx=0)
    (position_embeddings): Embedding(512, 768)
    (token_type_embeddings): Embedding(2, 768)
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (encoder): BertEncoder(
    (layer): ModuleList(
      (0-11): 12 x BertLayer(
        (attention): BertAttention(
          (self): BertSelfAttention(
            (query): Linear(in_features=768, out_features=768, bias=True)
            (key): Linear(in_features=768, out_features=768, bias=True)
            (value): Linear(in_features=768, out_features=768, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
```

```
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
)
```

Максимальный скор, полученный таким использованием, была достигнута при помощи модели catboost, и составляет 16.73.

После этого была применена Берта для классификации, обученная на твитах в соцсетях и частично дообученная на датасете. Она имела вид:

```
BertForSequenceClassification(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(119547, 768, padding_idx=0)
      (position_embeddings): Embedding(512, 768)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): BertEncoder(
      (layer): ModuleList(
        (0-11): 12 x BertLayer(
          (attention): BertAttention(
            (self): BertSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
```

```
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
    )
    (pooler): BertPooler(
      (dense): Linear(in_features=768, out_features=768, bias=True)
      (activation): Tanh()
    )
  )
  (dropout): Dropout(p=0.1, inplace=False)
  (classifier): Linear(in_features=768, out_features=3, bias=True)
)
```

С ней не получилось достигнуть высокой точности, **однако после перехода на deBERTav2 for Sequence Classification получилось достигнуть максимального скора среди всех моих итераций: 27.73**.

```
DebertaV2ForSequenceClassification(
  (deberta): DebertaV2Model(
    (embeddings): DebertaV2Embeddings(
      (word_embeddings): Embedding(128100, 768, padding_idx=0)
      (LayerNorm): LayerNorm((768,), eps=1e-07, elementwise_affine=True)
      (dropout): StableDropout()
    )
    (encoder): DebertaV2Encoder(
      (layer): ModuleList(
        (0-11): 12 x DebertaV2Layer(
          (attention): DebertaV2Attention(
            (self): DisentangledSelfAttention(
              (query_proj): Linear(in_features=768, out_features=768, bias=True)
              (key_proj): Linear(in_features=768, out_features=768, bias=True)
              (value_proj): Linear(in_features=768, out_features=768, bias=True)
              (pos_dropout): StableDropout()
              (dropout): StableDropout()
            )
            (output): DebertaV2SelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-07, elementwise_affine=True)
              (dropout): StableDropout()
            )
          )
          (intermediate): DebertaV2Intermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
```

```
            (intermediate_act_fn): GELUActivation()
          )
          (output): DebertaV2Output(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-07, elementwise_affine=True)
            (dropout): StableDropout()
          )
        )
      )
      (rel_embeddings): Embedding(512, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-07, elementwise_affine=True)
    )
  )
  (pooler): ContextPooler(
    (dense): Linear(in_features=768, out_features=768, bias=True)
    (dropout): StableDropout()
  )
  (classifier): Linear(in_features=768, out_features=3, bias=True)
  (dropout): StableDropout()
)
```

4 Ablation Study

Таким образом, лучше всего сработала модель DebertaV2Model, в случае, если бы мы разделяли модель и классификатор, то лучший результат показывал catboost.

5 Future Work + Conclusion

В дальнейшем было бы интересно подробно изучить влияние макропараметров при дообучении моделей, взятых с https://huggingface.co/ для нашей задачи - на подробное изучение, к сожалению, не хватило времени. Скорее всего, более короткое дообучение смогло бы повысить качество предсказаний.

Другой вариант упрощения задачи, но использования частично информации о сущностях - использования категориальной переменной *'entity_tag'*. Её можно было бы использовать в моделях *catboost и LogisticRegression* для повышения точности прогноза. Один из примеров логических зависимостей, почему это могло бы помочь - в новостях стараются не писать негативно про национальности вследствие неполиткорректности, поэтому процент негативных эмоциональных окрасок у национальностей в новостях ниже - это видно по тестовому датасету. Тем не менее, этот эксперимент в даной статье проделан не был.

Кроме того, автором не было произведено значительной предобработки датасета. Вероятно, некоторые итерации (например, расширение датасета за счёт перевода на английский и обратно) могли бы улучшить точность модели.

References

Birat Bade Shrestha and Bal Krishna Bal. 2020. Named-entity based sentiment analysis of Nepali news media texts. // *Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications*, P 114–120, Suzhou, China, December. Association for Computational Linguistics.

Anton Golubev and Natalia Lukashevich. 2021. Issledovanie modeley neironnykh setey tipa bert dlya analiza tonal'nosti tekstov na russkom yazyke.

N. Kosykh. 2022. Primenenie modeli distill'acii znanii bert dlya analiza nastroeniy teksta.

Anastasia Kotelnikova and N Kropanev. 2021. Bert dlya analiza tonal'nosti russkikh tekstov na primere kaggle russian news dataset. *Obschestvo. Nauka. Innovatsii.*