ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(СПбГУ)

Образовательная программа бакалавриата «Науки о данных»

Отчёт по практике

Учебная практика (научно-исследовательская работа)

Выполнил студент 4 курса бакалавриата
(группа 22.Б05-мкн)
Михеев Артём Антонович

Научный руководитель:
Кандидат физико-математических наук
Ершов Василий Алексеевич

Санкт-Петербург

Текстовые эмбеддинги финансовых новостей, обратимые по смыслу.

Содержание

Введение. Постановка задачи. Актуальность.

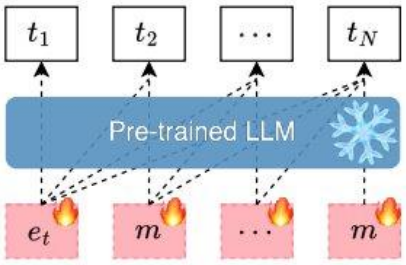
Обращать эмбеддинги токенов довольно просто. Находим эмбеддинг в матрице эмбеддингов и номер строчки является порядковым номером токена. Используя токенизатор, декодируем и получаем символы.

Предложений бывает очень много, и никакой памяти не хватит для хранения всех эмбедингов. Задача заключается в построении текстовых эмбеддингов с возможностью восстановления по смыслу. Обратить полностью не получится, так как какая-то часть информации теряется, но получить схожее по смыслу хотелось бы.

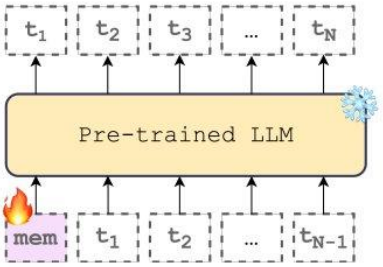
Предположим, мы проектируем мультимодальную модель для обработки данных разной природы. К примеру, такая модель может на основе числовых финансовых данных (временных рядов), новостного текста, вспомогательных таблиц предсказывать различные показатели: индексы фондового рынка, курс валют, возможен ли дефолт? Такие модели плохо интерпретируемые. Встает задача совместной генерации этих данных и текста-интерпретации – объяснения, почему произошло изменение этих финансовых показателей. Мы не можем генерировать токен за токеном, так как loss будет сосредотачиваться только на тексте, плохо генерируя показатели. Нам важно уловить общий смысл, получить одно представление (вектор) этого объяснения. Если получить такое представление, из которого можно сгенерировать текст, тогда для новой задачи нужно только научить получать это представление, а декодер уже есть.

Полезные статьи.

Авторы изучали возможность получения представления конкретного текста, то есть без encoder части. Вводились токены памяти (на схеме - mem) перед текстом и изучалось сжатие в эти токены. Основной акцент был на изучении локальности полученных представлений при разной инициализации. Авторегрессионный подход к генерации. И прямой, менее качественнный, но быстрый, только на основе эмбеддинга тескта, без предыдущих токенов. Был сделан вывод о возможности построения encoder части.



прямой



авторегрессионный

Формулировка задачи

Проверка гипотезы о возможности (и способах) конструирования такого эмбеддинга и изучения насколько можно сжимать тексты сохраняя смысл.

Основные результаты практики

Наша задача спроектировать и обучить как энкодер для получения одного представления, так более важная часть модели - decoder для смыслового восстановления.

Dataset.

Будем обучаться на финансовых новостях **Financial News Headlines Data с kaggle.**

Заголовки этих наборов данных, взятые с официальных сайтов CNBC, The Guardian и Reuters, отражают обзор экономики и фондового рынка США за каждый день за 2018-2020.

Пример:

U.S. economy faces significant risks, long road to recovery: IMF staff.

UBS, Morgan Stanley expected to lead Vodafone Tower IPO: sources.

As big U.S. banks let customers delay payments, loan losses remain unclear.

China says it will act to protect its interests after UK Huawei ban.

Всего 53370 заголовков. Делим данные на train и validation, validation size = 0.1.

Метрики.

Надо оценивать на сколько декодируемый текст похож на оригинал. Будем оценивать следующие метрики:

1)Совпадение слов.

ACCURACY. Сопоставляем токены по позициям.

2)Семантическая близость.

BERTScore.

Модель:`'microsoft/deberta-xlarge-mnli'`

- Получает эмбеддинги для каждого токена из двух предложений.
- Вычисляет семантическое соответствие токенов, сопоставляя каждый токен из output с ближайшим по смыслу токеном в input (и наоборот). O(T^2).
- Итоговые P, R, F1 — агрегированные значения сходства между токенами.

bert_score(**"The car was damaged in the accident."**, **"The vehicle got wrecked during the crash."**) = **0.91**

bert_score(**"The economy is growing rapidly."**, **'My dog loves running in the park."**) = **0.35**

*PARAPH-SIM***.**

Модель:`'sentence-transformers/paraphrase-mpnet-base-v2'`

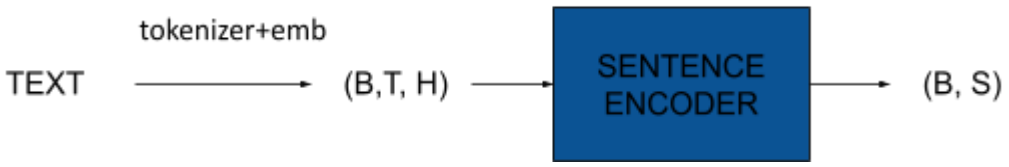
- Каждое предложение сжимается в один вектор фиксированной размерности 768.
- Вычисляется косинусное сходство между двумя векторами:
$$\cos\_\text{sim}(A, B) = \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

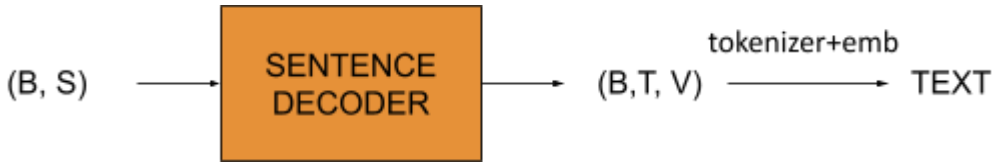
paraph_sim(**"The car was damaged in the accident."**, **"The vehicle got wrecked during the crash."**) = **0.821**

paraph_sim("**The economy is growing rapidly.", 'My dog loves running in the park.**") = **0.17**

Размерности.

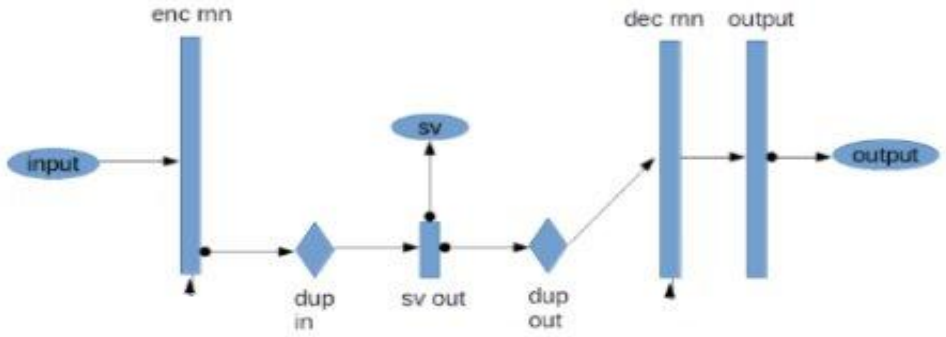
B - batch_size
T - длина предложения в токенах с padding для обучения по бачам.
H - размерность эмбеддингов токенов.
S - размерность эмбеддингов предлодений.
V - размер словаря из токенов.
На вход модели подаётся тензор размера (B, T, H). На выходе logits (B,T,V)





Архитектура моделей.

Прошлые подходы на основе RNN показали способность учиться, однако качество по метрикам было не очень.



Новые подходы на основе transformer архитектуры.

За основу encoder был взят ***FinBERT*** — это предварительно обученная модель для анализа настроений (сентимента) финансовых текстов.

На выходе ***FinBERT*** эмбеддинги для каждого токена и **cls** токен.

Агрегируем с обучаемыми весами выход ***FinBERT*** с помощью ***Attention pooling*** слоя в эмбеддинг предложения. Также для эксперимента подаем только **cls**.

Decoder обучаем с нуля двумя способами.

(1) За один проход. Decoder на основе transformer_encoder.
sent_emb: эмбеддинг предложения. Расширяем до T, учитывая позиции (прибавляем позиционные эмбеддинги) и подаем в decoder для self-attention.
Предсказываем все токены на основе ***sent_emb*.**
P(***tgt***[1:T] | ***sent_emb***)

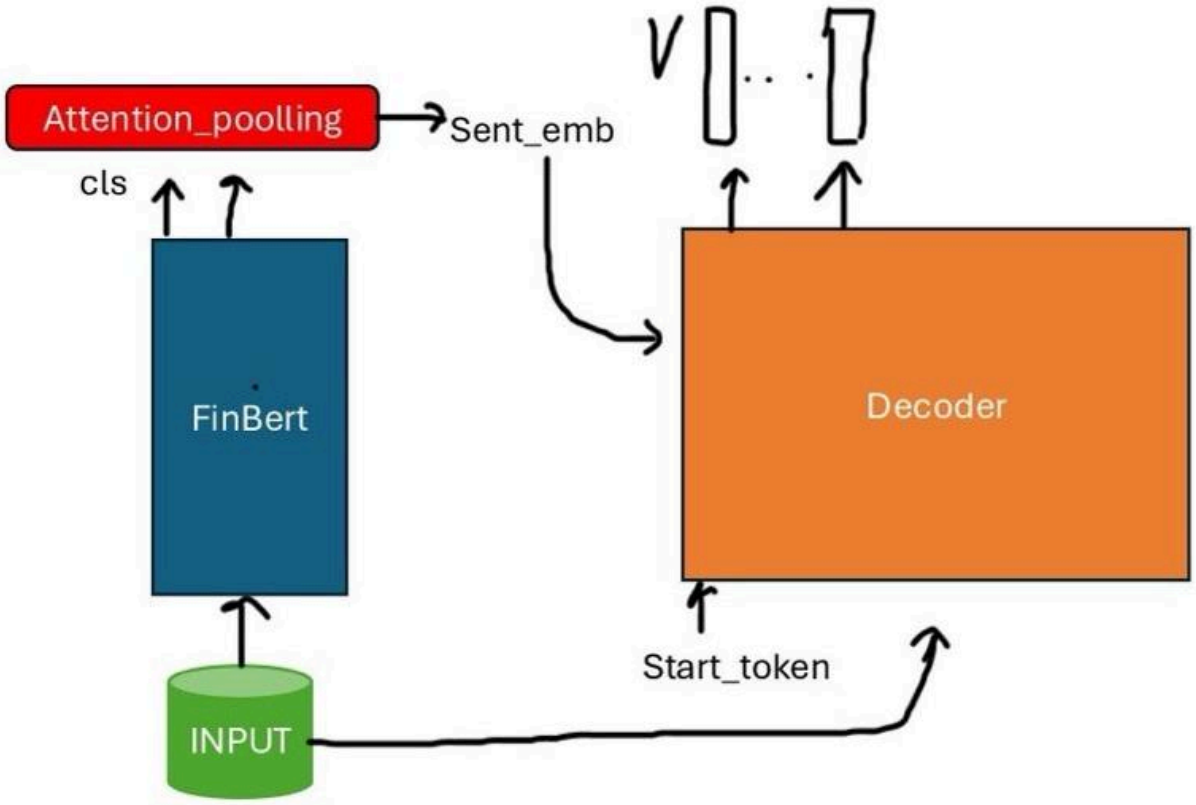
(2) Авторегрессионно. Decoder на основе transformer_decoder.
***sent_emb*:** эмбеддинг предложения. Подаем в decoder как вход для cross-attention.
tgt: токены входа. Подаем в decoder для masked self-attention и cross-attention.
Предсказываем следующий токен на основе предыдущих и sent_emb.
P(***tgt***[N+1] | ***tgt***[1:N], ***sent_emb***)

Attention Pooling

```
class AttentionPooling(nn.Module):
    def __init__(self, hidden_dim, output_dim):
        super().__init__()
        self.attn = nn.Linear(hidden_dim, 1)
        self.proj = nn.Sequential(
            nn.Linear(hidden_dim, output_dim),
            nn.GELU(),
            nn.LayerNorm(output_dim),
        )
    def forward(self, x, mask=None):
        attn_scores = self.attn(x).squeeze(-1)  # (B, T)
        if mask is not None:
            attn_scores = attn_scores.masked_fill(mask == 0, -1e9)
        attn_weights = torch.softmax(attn_scores, dim=1)  # (B, T)
        return torch.sum(attn_weights.unsqueeze(-1) * self.proj(x),
dim=1)  # (B, S)
```

Последний слой для получения ***logits.***

```
self.fc_out = nn.Sequential(
            nn.Dropout(0.5),
            nn.Linear(sent_dim, vocab_size)
        )
```



Обучение

Loss: ***cross_entropy_loss***
+ fine_tuning with ***sent_sim_loss***=cos(sent_emb(input), sent_emb(decoded))

optimizer: AdamW (`learning_rate=0.0001, weight_decay=0.01`)

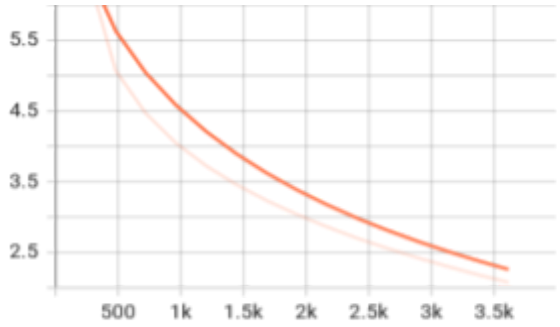
lr_scheduler: ReduceLROnPlateau (`mode='max', factor=0.1, min_lr=1e-5, patience=5`)

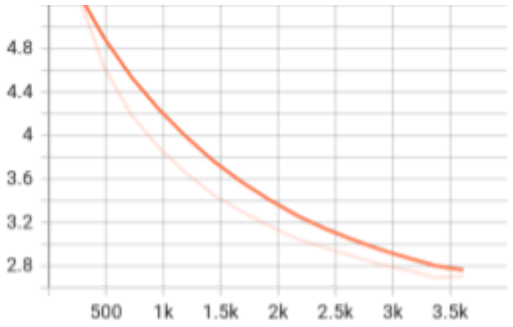
batch_size: 128-200

Графики

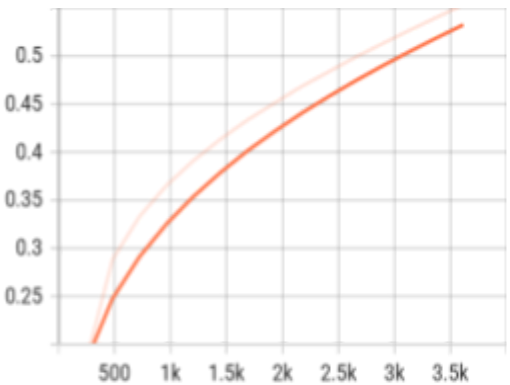
Train *VALIDATION*

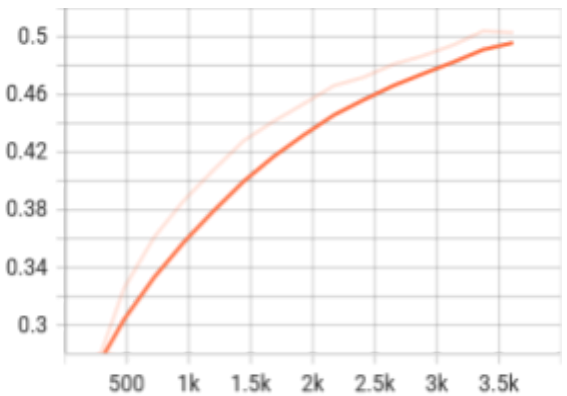
Loss



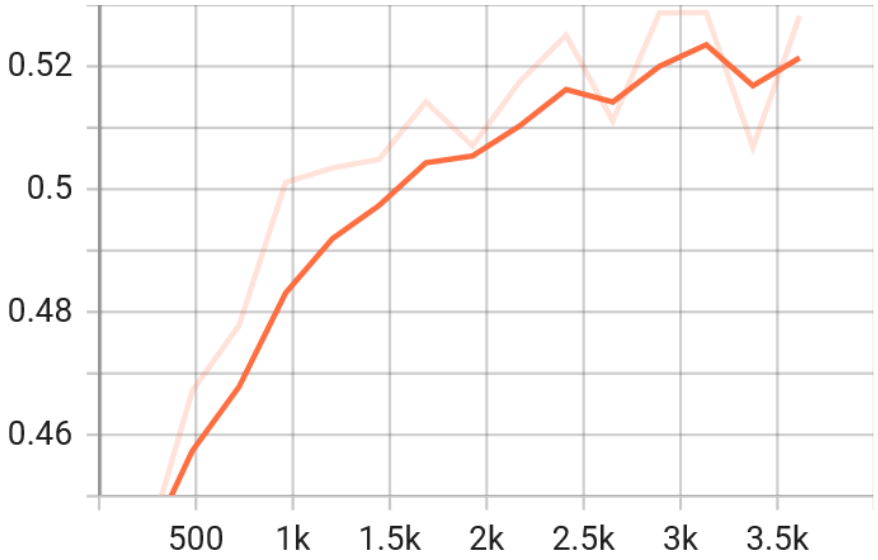


Accuracy

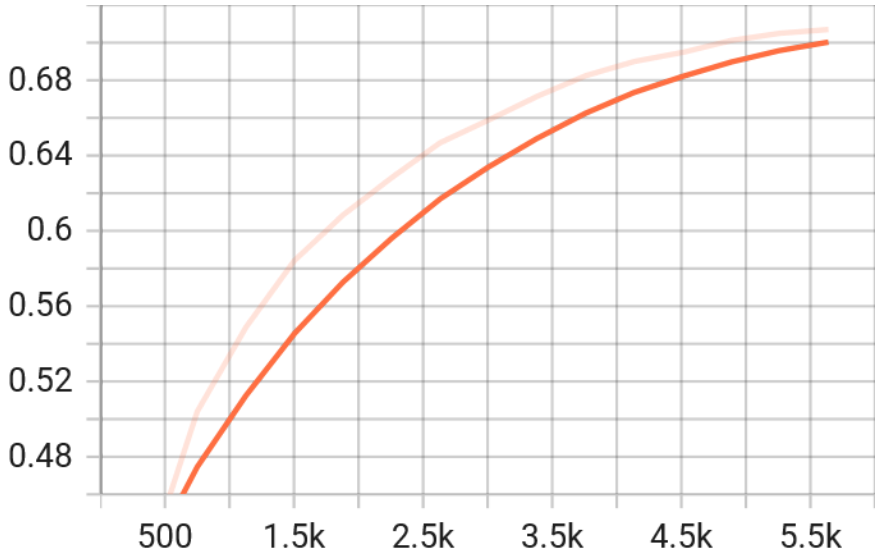




BERT-score



Top-5



Разные размерности

Зависимость метрик от архитектуры и размера эмбеддинга предложения.

model	*dim*	*Valid_accuracy*	*BERT_score*
За один проход	***1568***	***0.357***	
Авторегрессионно CLS	***768***	***0.406***	***0.58***
Авторегрессионно attention_pooliing	***768***	***0.502***	***0.65***
	2560	***0.552***	***0.675***

Аторегрессионные модели по качеству сильно лучше, однако медленнее.

Attention pooling увеличивает качеству по сравненению с ***cls*** эмбделлингом. Так же этот слой позволяет менять размерность ***sent_emb***.

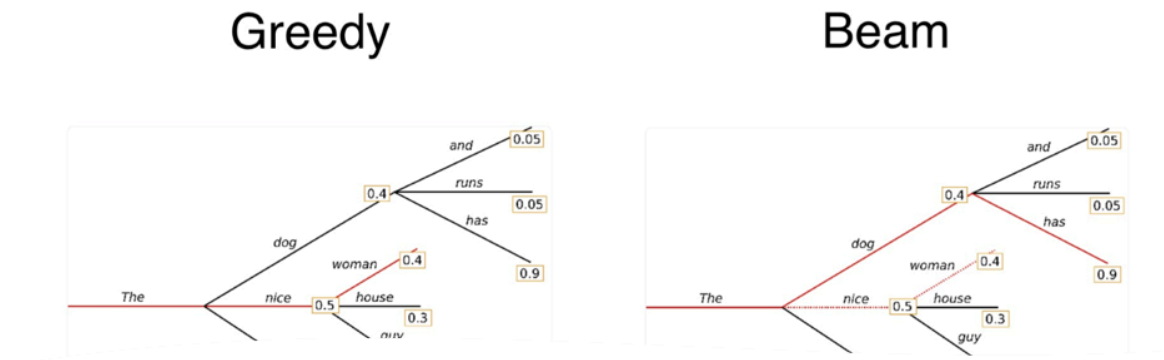
Увеличивая размерность вектора, метрики растут.

Как часто входные токены в первых k токенах по вероятности.

Top-k	*1*	*3*	*5*	*10*	*15*	*25*	*50*	*100*
Model_768	*0.50*	*0.63*	*0.68*	*0.73*	*0.75*	*0.78*	*0.81*	*0.84*
Model-2560	*0.55*	*0.68*	*0.72*	*0.77*	*0.79*	*0.81*	*0.84*	*0.86*

Заметно, что top-5 сильно лучше, чем top-1. С увеличением k скорость роста метрик сильно уменьшается. Возможно, что top-k для маленького k содержит синонимы или перефразы.

Генерация. GREEDY vs BEAM-SEARCH



Изменив алгоритм генерации с жадного на поиск наиболее вероятной последовательности, модель стала реже заикаться (повторять слова).

bert_score: 0.66 → 0.7

paraph_sim: 0.64 → 0.69

LOSS на уровне предложений

Было замечено, что cross_entropy loss переобучался на train с 20 эпохи. Поэтому были придуманы и реализованы loss на уровне предложений, а не токенов.

Cosine_similarity loss

Для исходного и восстановленого текста можно получить специальные эмбеддинги, и считать cosine_similarity между ними.

paraph_model:'sentence-transformers/paraphrase-mpnet-base-v2'
Выбор пал на paraph_model, используемую в вычислении метрики paraph_sim, так как она обучена получать эмбеддинги всего текста близкие у похожих по смыслу. Проще использовать свою же модель, но для разных по смыслу предложений cos_sim большой.

Сравнение.

	paraph_model	my_model
I enjoy reading books in my spare time.		
I like reading novels during my free time.	0.918	0.792
Python is a popular programming language.	0.162	0.544

Генерировать их распределения над токенами напрямую нельзя, не будет течь градиент.
Сэмплируем из logits с помощью gumbel trick.
distribution = F.gumbel_softmax(logits, tau=1.0, hard=True)
Straight-Through Estimator (STE) distribution=one_hot + (soft - soft.detach())
gen = torch.matmul(distribution , emb_matrix) - эмбеддинги токенов

Подаем в paraph_model gen вместо токенов.
emb_inp = paraph_model(input_text)
emb_gen = paraph_model(inputs_embeds =gen)

Loss = cosine_similarity(emb_inp, emb_gen)
Этот loss оказался очень не стабильным. Предлагается использовать его как добавку к базовому loss с небольшим коэфициентом в качестве регуляризатора.

Loss на перефразах

Наша задача восстановить по смыслу, можем это учитывать в loss. И напрямую обучать восстанавливать перефразы.

gen_paraphrase_model-"ramsrigouthamg/t5-large-paraphraser-diverse-high-quality"
С помощью gen_paraphrase_model создаем Paraphrase Dataset, по 5 перефраз на пример из train.
Для разнообразной генерации можно изменять параметры.
Параметры генерации: top_k=120, top_p=0.9, temperature=1.7

Пример из Paraphrase Dataset
base:
Weak sales growth hits American Eagle outlook; shares slide

paraphrase1
The American Eagle's outlook is marred by poor sales growth; the stock plummets.

paraphrase2:
The American Eagle's prospects are harmed by poor sales growth; the stock plummets.

paraphrase3:
The American Eagle's prospects suffer as a result of poor sales growth; the stock plummets.

loss = ce_loss + min(ce_loss1, …, ce_loss5),
ce_loss{i} - cross_entripy loss для i-ой перефразы
Это поможет восстанавливать именно перефразы.

Этот loss не хуже базового, менее склонен к переобучению.

Изучение полученных эмбеддингов

Задача изучить наши эмбеддинги на локальность, чтобы при неточной их генерации, восстановление ухудшалось несильно.
Добавить к эмбеддингам нормальный шум N(0, sigma), измерить качество восстановления.
Bert_score(sigma = 0.1) ~ Bert_score(sigma = 0) Bert_score(sigma = 1) < Bert_score(sigma = 0)*0.9

Примеры восстановления.

INPUT	*DECODED*
consumer giants spurn risks to chase online subscribers	*consumer giants spurn consumer risks to online subscribers*
united technologies wins $ 2. 2 billion u. s. defense contract : pentagon	*united technologies wins $ 2. 2 billion u. s. defense contract : pentagon*
australia watchdog says vodafone misled customers over digital purchases	*australia watchdog says hsbc over wireless firms misled customers payments*
bayer says monsanto likely kept files on influential people across europe	*bayer says monsanto likely put on some countries sick people in europe*
wall street rises on economic data, easing trade worries	*wall street rises on trade worries, fed data data*

malaysia regulator to probe if airasia broke rules in airbus deals	***malaysia regulator to probe airasia in regulators if boeing deals***

(1-2) целиком восстанавливаются. В (3) компания Vodofone восстановлена как фирма, занимающаяся беспроводной связью. Payments почти синоним purchases. (6) boeing - airbus. Остальные примеры имеют похожие слова.

Заключение

Основные цели проекта были достигнуты. Была показана возможность построения текстого эмбеддинга с возможностью частичного восстановления по смыслу с BERTScore=0.7. Были исследованы и реализованы различные архитектуры, а также придуманы специальные функции потерь, напрямую отражающие идею восстановления текста похожего по смыслу.

Дальнейшее изучение:
Изучить эмбеддинги на локальность.
Использовать предобученные *LLM(Llama, Pythia)* как более мощные decoder-архитектуры.
Попробовать для нашей задачи rnn-подобные архитектуры*(Mamba, Gated Delta Net)*. Они основаны на состоянии всего текста, и, инициализируя модель состоянием нашего текста, можно попросить повторить, что было до. Проблема в том, что это состояние имеет размерность dim x dim >> dim. Это предстоит изучить.

Данный подход открывает новые подходы к совместной генерации мультимодальных моделях. Так для объяснения изменения финансовых показателей можно предсказывать один вектор. А наш декодер сгенерирует текст, сохранивший общий смысл.
Список использованных литературных источников
и информационных материалов.

[1] Cramming 1568 Tokens into a Single Vector and Back Again: Exploring the Limits of Embedding Space Capacity

[2] Exploring the Latent Capacity of LLMs for One-Step Text Generation

Репозиторий проекта.

Github: *sber_practice*