НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Сокращение описаний товаров без потери релевантной информации

Отчёт по исследовательскому курсовому проекту

Автор: Альтшулер Э. Д., группа БПИ233
Руководитель: Соколов А. П., доцент каф. ПМИ, НИУ ВШЭ (Нижний Новгород)

Москва, 2026

2. Предметная область

Контекст задачи

- Яндекс.Маркет — одна из крупнейших маркетплейс-платформ в России; каталог насчитывает **сотни миллионов** товарных предложений
- Трансформерные языковые модели применяются в ключевых продуктовых задачах: **поиск**, **ранжирование**, **рекомендации**, **матчинг** предложений
- На вход моделей поступают: название товара, структурированные характеристики, **текстовое описание**
- Текстовые описания поставщиков содержат значительную долю маркетинговых конструкций: эпитеты («потрясающий», «уникальный»), повторения, призывы к покупке — информационная плотность таких фрагментов низка

Вычислительная проблема

Сложность операции self-attention:

$$O(n^2)$$

по времени и памяти, где n — длина входной последовательности

Обработка маркетингового шума расходует вычислительный ресурс без полезного выхода. При масштабе в сотни миллионов карточек сокращение длины входа на **30–40%** обеспечивает кратную экономию вычислений.

Архитектурные модификации (Longformer, BigBird) требуют переобучения базовых моделей. **Сжатие входа** применимо к существующим системам без их изменения — встраивается на уровне препроцессинга.

3. Актуальность

Вычислительные затраты

Пусть каталог содержит N товаров, каждое описание длиной n токенов. Стоимость инференса одного токена — c. Суммарная стоимость обработки:

$$C = N \cdot n \cdot c$$

При сокращении длины описания до $n' = \alpha n$, где $\alpha < 1$:

$$C' = N \cdot \alpha n \cdot c = \alpha C$$

При $N = 10^8$, $n = 1000$, $\alpha = 0.56$ (результат работы) экономия составляет **44%** вычислительных затрат.

Обоснование подхода

Прямое применение LLM-компрессора в проде требует на каждый запрос $O(n^2)$ операций тяжёлой модели. При $N = 10^8$ это экономически неприемлемо.

Дистилляция переносит поведение учителя в компактную модель:

$$\theta^* = \arg\min_\theta \mathcal{L}(f_\theta, f_{\text{teacher}})$$

где f_{teacher} — YandexGPT, f_θ — BERT $\approx$ 0.1B. После обучения инференс ученика: $O(n)$ — линейный от длины входа.

Отсутствие публичных методов для доменной адаптации к русскоязычным товарным описаниям обосновывает самостоятельную разработку.

4. Цель и задачи

Цель: построить метод автоматического экстрактивного сжатия товарных описаний, ускоряющий трансформерные модели без ухудшения качества downstream-задач

1. Проанализировать существующие методы сжатия промптов
2. Собрать датасет через разметку YandexGPT
3. Обучить токен-классификатор на базе BERT (0.1B параметров)
4. Разработать методику оценки качества через LLM as Judge
5. Провести ablation-эксперимент со случайным удалением слов

5. Анализ существующих подходов

Метод	Идея	Ограничение для нашей задачи
LLMLingua (2023)	Оценка важности токена по перплексии; удаление «предсказуемых» токенов	**Доменный сдвиг:** перплексия общеязыковой модели не отражает информативность для товара
LongLLMLingua (2024)	Учёт зависимости от запроса (question-aware compression)	Нет товарных данных; заточен под QA-задачи
LLMLingua-2 (2024)	Дистилляция GPT-4 → компактный XLM-R с учителем	Обучен на reasoning; не покрывает стилистику товарных описаний
Абстрактивное сжатие	Перефразирование через LLM	**Риск искажения фактов** (цифры, бренды); авторегрессия дорога на масштабе 100M+ карточек

Вывод: нужен специализированный **экстрактивный** метод, обученный на доменных данных

6. Архитектура решения: дистилляция знаний

Учитель — YandexGPT

- Промпт с «железными правилами» и 3 few-shot примерами
- Принцип подпоследовательности — слова только из оригинала
- Пары: исходное описание ⟶ сжатое
- Выравнивание через LCS ⟶ бинарная метка на каждый токен

92.6% пар прошли автоматическую валидацию

⟶

Ученик — BERT 0.1B

- Предобученная BERT-модель + линейный слой
- Взвешенный BCE, оптимизатор AdamW
- LR = 1×10^{-5} · batch = 1 280 · iters = 10 000
- Инференс: $P(\text{keep}|t_i) > \tau \longrightarrow$ сжатый текст

В проде: в 100× дешевле учителя

7. Промпт для YandexGPT

Системная роль:

«эксперт-лингвист по экстрактивному сжатию»

Железные правила:

1. **Подпоследовательность** — слова только из оригинала
2. **Запрет изменения форм** слов
3. **СОХРАНЯТЬ:** бренды, числа, единицы (см/кг/Вт/ГБ), материалы
4. **УДАЛЯТЬ:** эпитеты, маркетинговая вода, предлоги-союзы

Few-shot примеры (3 шт.):

Оригинал: Этот потрясающий смартфон Samsung Galaxy S23 оснащён невероятным экраном с диагональю 6.1 дюйма…

Сжатый: смартфон Samsung Galaxy S23 экраном 6.1 дюйма процессором

Эффект «железных правил»:

Без них **9%** ответов YandexGPT нарушали принцип подпоследовательности — добавляли новые слова или меняли падежи. Такие пары непригодны для обучения.

После добавления блока правил нарушений стало **< 1%**.

9% → **< 1%**

8. Сбор и балансировка датасета

Валидация пар:

Этап	Осталось
Исходный пул	100.0%
Проверка подпоследовательности	95.7%
Фильтр степени сжатия	93.8%
Дедупликация по MD5	**92.6%**

Балансировка по 4 бинам:

Бин	Токенов
bin_0_256	< 256
bin_256_512	256–512
bin_512_1024	512–1024
bin_1024_plus	≥ 1024

Случайная перестановка → усечение до размера минимального бина

71 672 878
обучающих примеров

11 837
валидационных примеров

9. Архитектура модели-ученика

Токены $t_1, t_2, ..., t_n$ описания товара

↓

BERT encoder — предобученный, 0.1B параметров

↓

Контекстные представления $h_1, ..., h_n \in \mathbb{R}^d$

↓

Linear head: $\hat{y}_i = \sigma(Wh_i + b)$

↓

Вероятности $P(\text{keep} \mid t_i)$ для каждого токена

Параметры обучения:

Параметр	Значение
Базовая модель	BERT 0.1B
Оптимизатор	AdamW
Learning rate	1×10^{-5}
Global batch size	1 280
Число итераций	10 000
Функция потерь	Взвешенный BCE

Модель принимает до **30 000 токенов** — описания подаются целиком, без нарезки.

10. Двухстадийная схема обучения модели-ученика

Стадия 1: обучение головы

BERT Encoder — заморожен → **Linear Head** — обучается

- Веса BERT-энкодера **зафиксированы** — предобученные представления сохраняются
- Обучается только линейный классификационный слой поверх энкодера
- Цель — адаптировать голову к задаче без дестабилизации предобученных весов

→

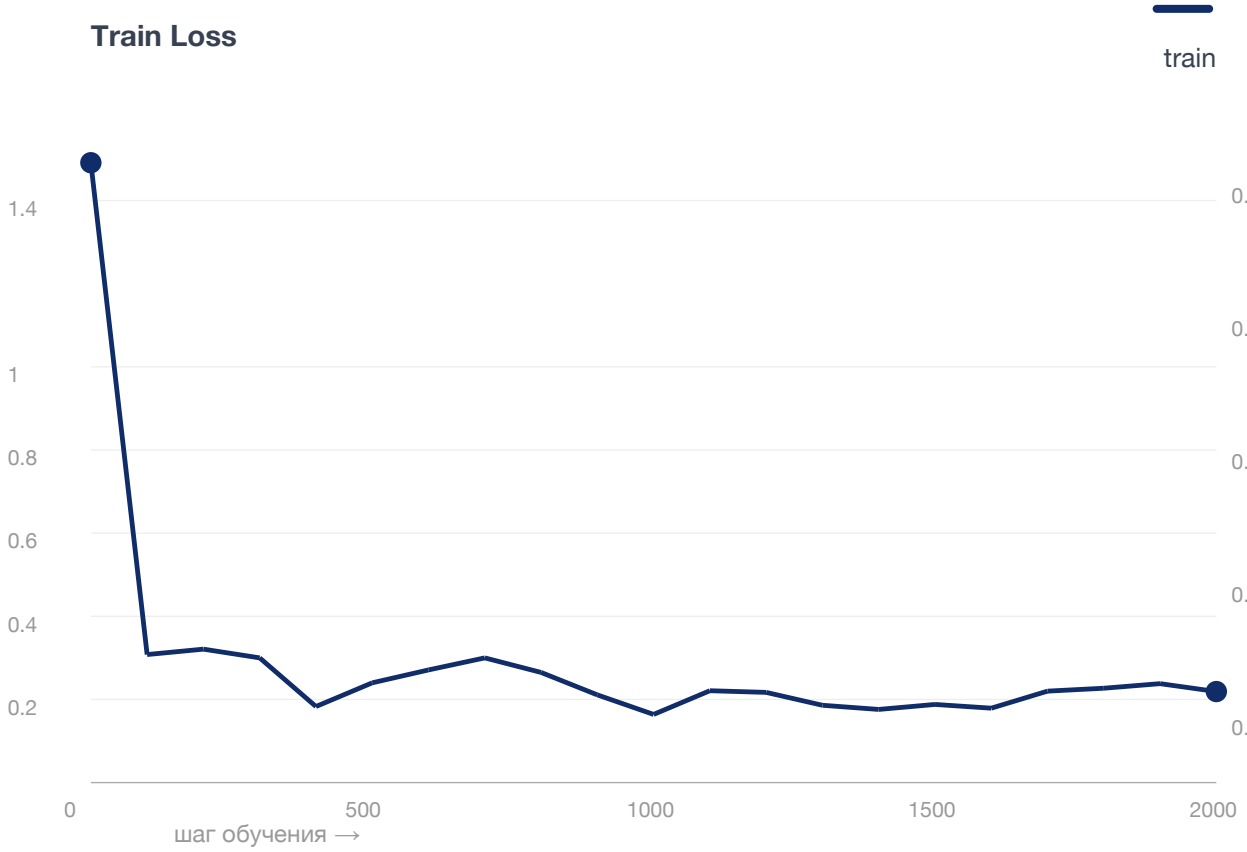
Стадия 2: полное обучение

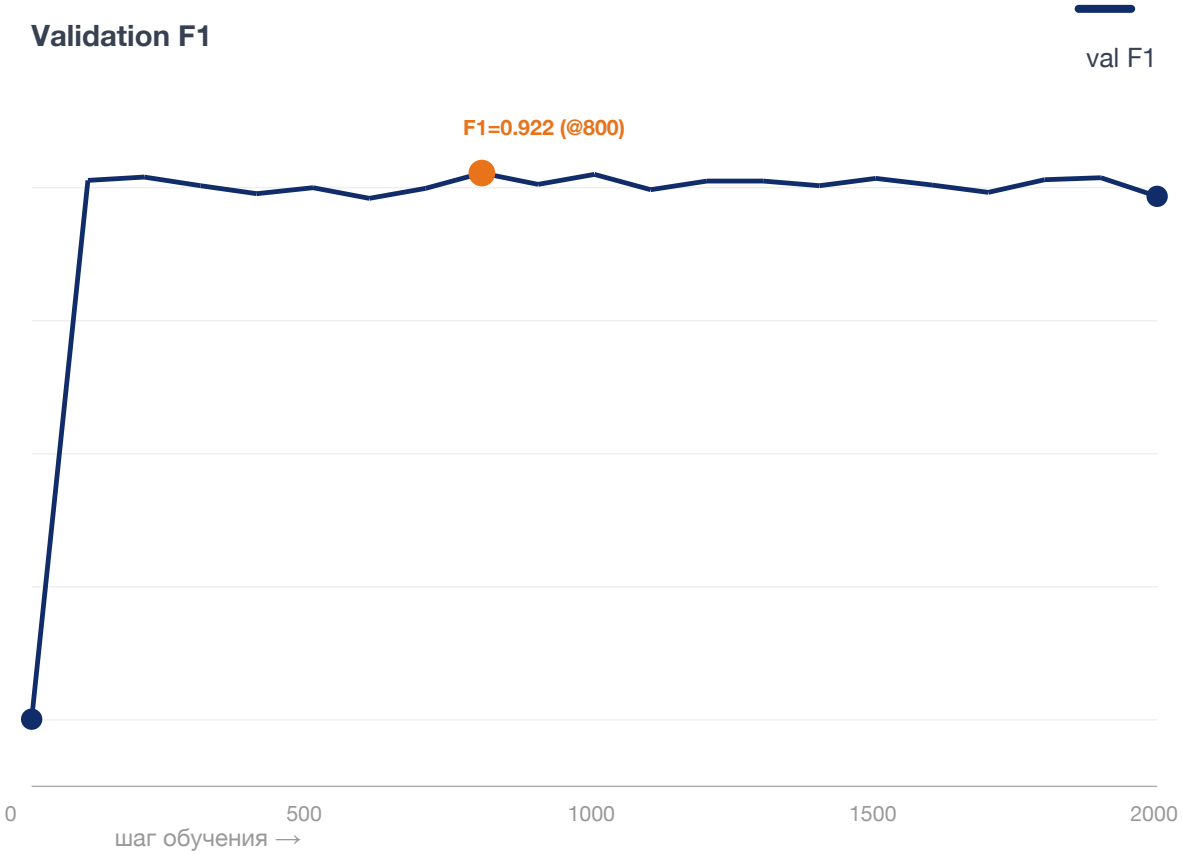
BERT Encoder — обучается → **Linear Head** — обучается

- Размораживается **весь энкодер**
- Совместное обучение BERT + head на товарных описаниях
- LR = 1×10^{-5}, batch = 1 280, iters = 10 000
- Цель — адаптировать контекстные представления к доменной лексике

Обоснование: модель предобучена на общих текстах. Прямое полное обучение может разрушить полезные представления. Стадия 1 формирует устойчивую инициализацию головы; стадия 2 адаптирует всю сеть к предметной области.

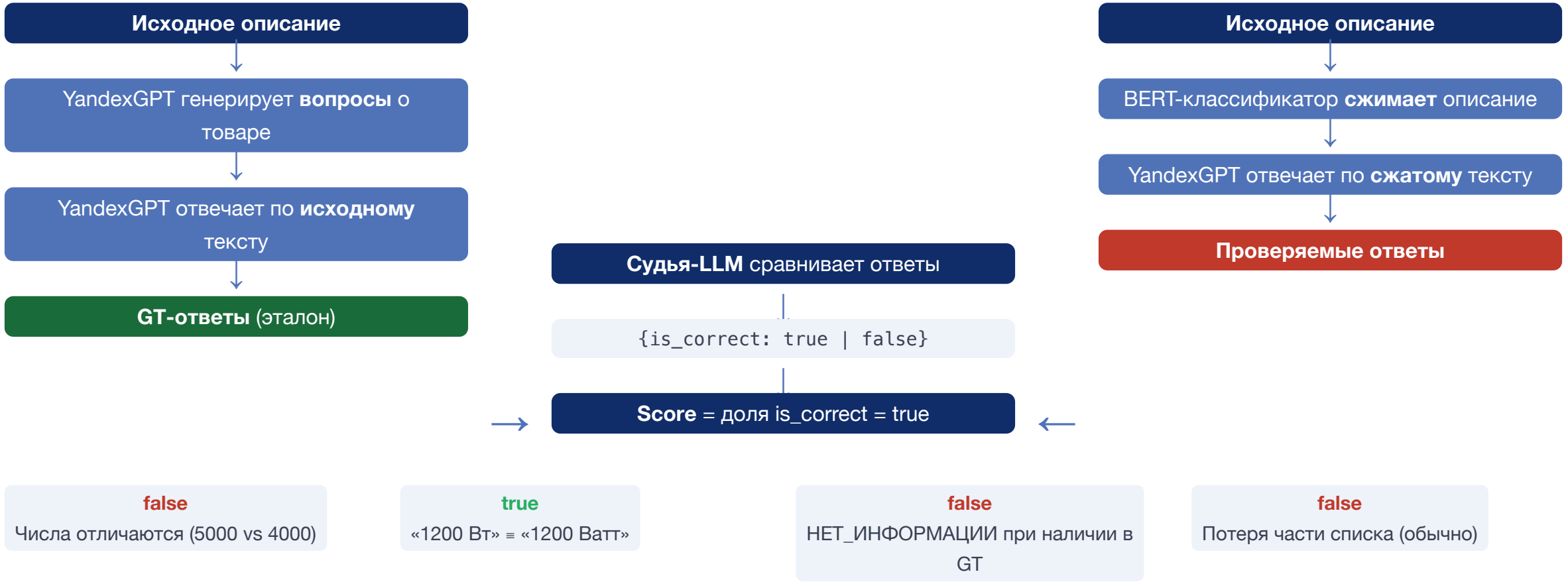
11. Метрики обучения





Ограничение метрики F1: высокое значение F1 (≈ 0.92) достигается за счёт правильной классификации частотных «нулевых» токенов (союзов, предлогов). Модель может удалить «Samsung Galaxy S23» или «6.1 дюйма» — F1 практически не снизится, тогда как информация о товаре будет безвозвратно потеряна. **Требуется независимая метрика сохранности фактической информации.**

12. Методология оценки: LLM as Judge



false
Числа отличаются (5000 vs 4000)

true
«1200 Вт» ≡ «1200 Ватт»

false
НЕТ_ИНФОРМАЦИИ при наличии в GT

false
Потеря части списка (обычно)

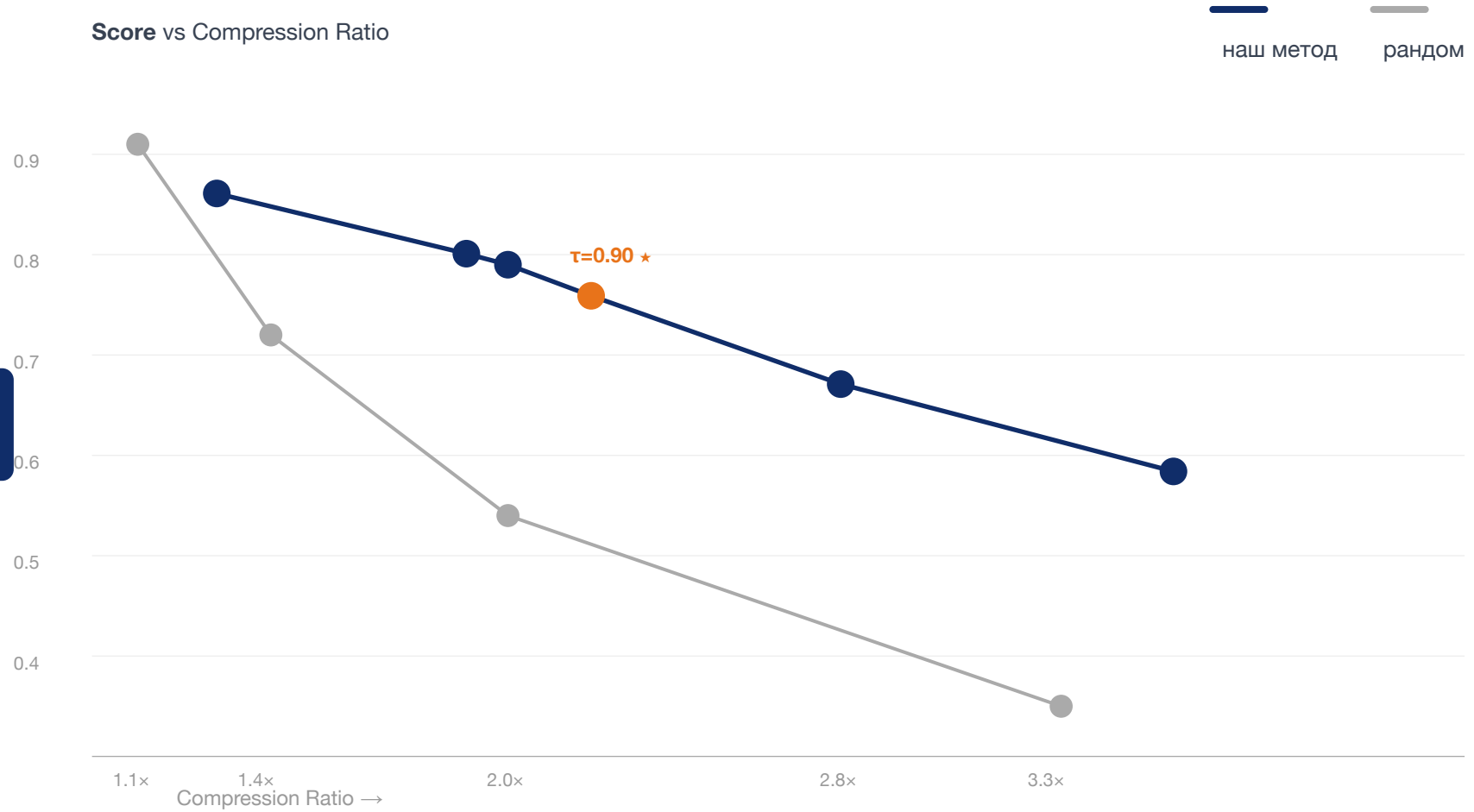
13. Результаты: метод vs. случайное удаление

Слов	Рандом	Метод	Δ
30%	0.35	**0.584**	**+0.23**
50%	0.54	**0.759** ⋆	**+0.22**
54%	≈0.58	**0.801**	**+0.22**
70%	0.72	**0.861**	**+0.14**

⋆ рабочая точка τ = 0.90

+14–23 п.п.

при любом уровне сжатия



14. Основные результаты

Достигнутые результаты:

- Метод дистилляции YandexGPT → BERT 0.1B разработан и проверен
- Датасет: **71.7M** примеров, балансировка по 4 бинам длины
- τ = 0.90: сжатие **2.2×**, Score **0.759** (80% от базлайна)
- Превосходство над случайным удалением: **+14–23 п.п.**

Новизна:

- Адаптация методов сжатия промптов к товарному домену
- Промпт с гарантией принципа подпоследовательности на уровне слов
- Методика LLM as Judge с бинарным JSON-вердиктом

Практическая значимость: применимо в продуктовых системах поиска, ранжирования и матчинга на масштабе сотен миллионов карточек

15. Направления дальнейшей работы

Улучшение качества модели

- **Улучшение архитектуры:** переход на более мощные энкодеры с увеличенным контекстным окном; добавление механизмов учёта позиционных зависимостей между токенами
- **Дополнительные функции потерь:** помимо BCE, введение регуляризации на сохранность числовых значений и named entities; contrastive loss для разделения информативных/шумовых токенов
- **Пер-категорийные классификаторы** или включение категории товара как дополнительного входного признака

Улучшение данных и применимости

- **Фильтрация обучающих данных:** более строгая валидация синтетической разметки; очистка от шумных примеров; балансировка по типам удаляемых токенов (эпитеты vs. структурные элементы)
- **Увеличение объёма данных:** расширение обучающей выборки, добавление описаний из смежных доменов и других языков
- **Учёт структурированных характеристик** при формировании датасета и инференсе — потенциальный источник дополнительного сжатия без потери качества

16. Список использованных источников

1. Vaswani A. et al. Attention Is All You Need. arXiv:1706.03762 — 2017
2. Devlin J. et al. BERT: Pre-training of Deep Bidirectional Transformers. arXiv:1810.04805 — 2018
3. Hinton G. et al. Distilling the Knowledge in a Neural Network. arXiv:1503.02531 — NIPS Workshop, 2015
4. Jiang H. et al. LLMLingua: Compressing Prompts for Accelerated Inference of LLMs. arXiv:2310.05736 — EMNLP 2023
5. Jiang H. et al. LongLLMLingua: Accelerating and Enhancing LLMs in Long Context Scenarios. arXiv:2310.06839 — ACL 2024
6. Pan Z. et al. LLMLingua-2: Data Distillation for Efficient and Faithful Prompt Compression. arXiv:2403.12968 — Findings ACL 2024
7. Zheng L. et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. arXiv:2306.05685 — NeurIPS 2023
8. Hirschberg D. S. Algorithms for the Longest Common Subsequence Problem. Journal of the ACM, Vol. 24, No. 4, 1977
9. Яндекс. YandexGPT: описание модели [Электронный ресурс]. — URL: ya.ru/ai/gpt — 2024