

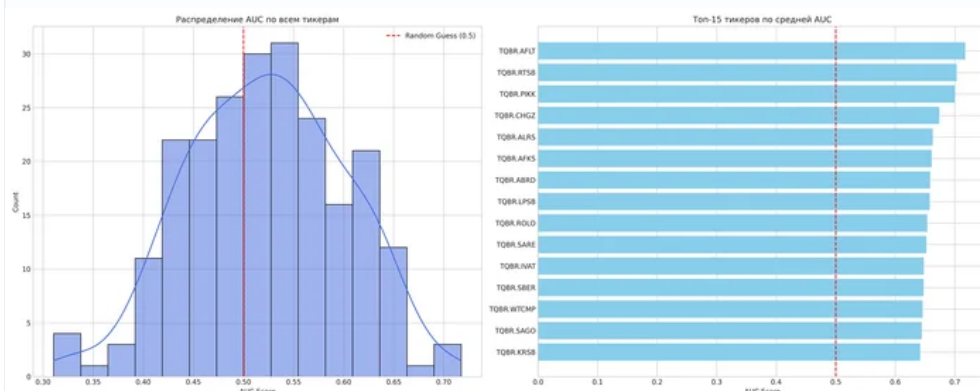
[Горячее](#) [Лучшее](#) [Свежее](#) ...Я съела
носок[Войти](#)[empenoso](#) [Искусственный интеллект](#)

...

Если трейдеры говорят словами, почему нейросеть не может? Мой эксперимент

[🕒 3 месяца назад](#) [👁 3.5K](#)

Представьте опытного трейдера: наверняка он не говорит котировками и не рассказывает про индикаторы — он просто говорит «сильный тренд», «пробой уровня» или «ложный отскок». Для него график это язык: свечи, объёмы и уровни складываются в понятные фразы о том, что сейчас происходит на рынке. Именно от этой человеческой интуиции я и отталкивался в своём эксперименте.



Мои результаты, о них ниже

Идея была такая: а что, если научить искусственный интеллект понимать этот язык? Не подавать модели сырые числа, а переводить бары и объёмы в текстовые описания наблюдаемых паттернов и кормить ими языковую модель. Гипотеза была что в тексте уже будет содержаться достаточно данных, чтобы модель научилась связывать недавнюю торговую историю с тем, пойдёт ли цена вверх на следующий день.

Инструмент эксперимента — модель [distilbert-base-uncased](#) с Hugging Face и это облегчённая, быстрая версия [BERT](#) для понимания языка. Мне показалось это практичным выбором для прототипа — позволяет быстро проверять разные способы текстовой разметки без гигантских ресурсов. Цель была чёткая: по текстовому описанию недавней истории торгов предсказать рост цены на следующий день.

Но это исследование моя попытка представления рыночных данных как языка, а не попытка сразу создать алгоритм для автотрейдинга. Ещё важно: это мой личный эксперимент, проведённый одним человеком и выполненный однократно. Результаты дали интересные наблюдения.

Расскажу, как происходила разметка графиков в текст, какие шаблоны сработали лучше и какие метрики использовались. Также отмечу ограничения подхода и идеи для повторных экспериментов.

Войти

[Войти](#)[Создать аккаунт](#)[Забыли пароль?](#)[или продолжите с](#)[Войти с Яндекс ID](#)[Войти через VK ID](#)[Промокоды](#)[Работа](#)[Курсы](#)[Реклама](#)[Игры](#)[Пополнение Steam](#)

А ещё весь код уже на GitHub.

Для трейдеров и аналитиков: суть и результаты

Для модели котировки это просто цифры без контекста. Она не знает, что вверх — хорошо, а вниз тревожный сигнал. Я переводил ряды котировок в текст. Каждые 10 дней торгов превращались в короткое описание, как если бы трейдер рассказывал что-то своему коллеге.

В основе лежали три признака:

- Краткосрочный тренд (3 дня) — рост, падение или боковик;
- Среднесрочный контекст (7 дней) — подтверждает ли он текущее движение;
- Моментум и объём — поддерживают ли рост деньги: идёт ли рост на растущих объёмах или затухает.

Если цена три дня растёт, объёмы увеличиваются, и цена близка к сопротивлению, строка выглядела так:

price rising strongly, volume increasing, near resistance.

Эти описания читала модель DistilBERT. Модель не видела графиков, только текст — и должна был сказать, приведёт ли ситуация к росту или падению. Так модель «училась понимать» то, что трейдеры выражают словами.

```
mike@linux-pc: ~/SynologyProjects/2025_10_предсказание котировок
Обучение моделей по тикерам: 100% [██████████] 20/20 [02:50<00:00, 0.53s/it]
{'train_runtime': 2.1447, 'train_samples_per_second': 234.998, 'train_steps_per_second': 7.46, 'train_loss': 0.6814438104629517, 'epoch': 2.0}
✓ TQBR.BSPBP: AUC = 0.5159 ± 0.0136

Результаты сохранены в: results/experiment_results_20251011_125928.json

=====
АНАЛИЗ РЕЗУЛЬТАТОВ
=====

✓ Общая производительность (по 20 тикерам):
  ticker  auc  accuracy  f1
TQBR.AFLT 0.7175 0.5556 0.0000
TQBR.ALRS 0.6629 0.4603 0.0000
TQBR.AFKS 0.6611 0.7143 0.0000
TQBR.ABRO 0.6584 0.6984 0.0000
TQBR.ARSA 0.6251 0.5397 0.0000
TQBR.ASSB 0.6099 0.6190 0.0000
TQBR.APTK 0.5944 0.6032 0.0000
TQBR.BISVP 0.5805 0.5714 0.0000
TQBR.ASTR 0.5606 0.4921 0.0000
TQBR.ABIO 0.5451 0.5397 0.0000
TQBR.AMEZ 0.5355 0.4762 0.0000
TQBR.AVAN 0.5208 0.7143 0.0000
TQBR.BSPBP 0.5159 0.5556 0.0000
TQBR.BELU 0.5158 0.5397 0.0000
TQBR.AQUA 0.4813 0.6190 0.0000
TQBR.BRZL 0.4712 0.6825 0.0000
TQBR.BANE 0.4603 0.5079 0.0000
TQBR.BLNG 0.4454 0.4921 0.6444
TQBR.AKRN 0.4421 0.4444 0.0000
TQBR.BANEP 0.4324 0.4444 0.3462

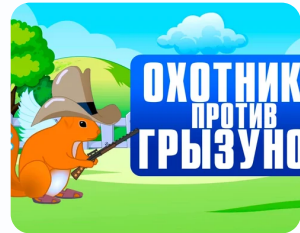
=====
  ACCURACY: Mean=0.5635, Std=0.0881, Median=0.5476
  PRECISION: Mean=0.0400, Std=0.1264, Median=0.0000
  RECALL: Mean=0.0679, Std=0.2278, Median=0.0000
  F1: Mean=0.0495, Std=0.1599, Median=0.0000
  AUC: Mean=0.5518, Std=0.0846, Median=0.5403

Визуализации сохранены в: results/analysis_20251011_125928.png
#####
### Эксперимент завершен. Результаты в папке ./results
#####
mike@linux-pc:~/SynologyProjects/2025_10_предсказание котировок$
```

Результат 20 бумаг в консоли

Как измерить, понимает ли она рынок

Пикабу Игры
+1000 бесплатных
онлайн игр



Охотники против грызунов

Мультиплеер, Аркады, 2D

Играть

Топ прошлой недели

- Animalrescued
41 пост
- a.lisik
3 поста
- Oskanov
8 постов

[Посмотреть весь топ](#)

Лучшие посты недели

Простая точность (ассигасу) ничего не говорит: можно всё время предсказывать падение и быть правым на 60%.

Поэтому я использовал **AUC (Area Under Curve)** — показывает, насколько хорошо модель отличает ситуации, после которых цена действительно росла, от тех, после которых она падала:

- $AUC = 1.0$: идеальная модель, которая никогда не ошибается.
- $AUC = 0.5$: бесполезная модель, ее предсказания равносильны подбрасыванию монетки.
- $AUC > 0.5$: модель работает лучше, чем случайное угадывание. Чем ближе к 1.0, тем лучше.
- $AUC < 0.5$: модель работает хуже случайного угадывания.

Эксперимент на всех акциях Московской биржи

Результат 227 бумаг в консоли

Я протестировал более 200 акций с Московской биржи. **Средний результат по всем бумагам — $AUC \approx 0.53$** , что немного лучше случайного угадывания.

Лучшие случаи:

- AFLT — 0.72
- RTSB — 0.70
- PIKK — 0.70
- CHGZ — 0.67
- AFKS — 0.66

Рассылка Пикабу:
отправляем самые
рейтинговые материалы за 7
дней 🔥

Укажи

Подписаться

Нажимая «Подписаться»,
я даю согласие на **обработку
данных** и **условия почтовых
рассылок**.

Помощь	Правила
Кодекс Пикабу	соцсети
Команда	О
Пикабу	рекомендация
Моб.	х
приложение	О компании

Промокоды Биг Гик
Промокоды Lamoda
Промокоды МВидео
Промокоды Яндекс Маркет
Промокоды Пятерочка
Промокоды Aroma Butik
Промокоды Яндекс
Путешествия
Промокоды Яндекс Еда
Постила
Футбол сегодня



Худшие:

- PLZL — 0.33
- VJGZP — 0.33
- CHMF — 0.36
- ETLN — 0.38
- LSNG — 0.39

Разброс большой: одни бумаги ведут себя предсказуемо, другие — как шум.

Что это значит для трейдера

С практической стороны — **торговать по такой схеме нельзя**. Даже при AUC 0.6 предсказательная сила слишком слаба, чтобы покрыть комиссии. Однако сам факт, что модель хоть немного «чувствует» структуру графика без чисел и свечей, уже интересен.

Эксперимент показал: график можно описать словами, и языковая модель способна уловить логику движения — пусть пока не точно.

Нагрузка на GPU

Для технических специалистов

Проект реализован на стеке Python + PyTorch + Hugging Face Transformers с изоляцией через Docker. Контейнер собирается на базе `pytorch/pytorch:2.7.0-cuda12.8-cudnn9-devel` для совместимости с современными GPU.

Модель дообучалась (fine-tuning) на базе DistilBERT, предобученной на английском тексте (distilbert-base-uncased). То есть — это классическая дообучаемая голова классификации (num_labels=2) поверх всего BERT-тела. Fine-tuning проходил end-to-end, то есть обучались все слои, не только голова.

Все слои разморожены. Базовая часть DistilBERT не замораживалась. Значит, fine-tuning шёл по всей модели (весам encoder'a + классификационной голове).

Конвейер данных строится в три этапа. Пакетная загрузка: метод load_all_data() читает все файлы котировок за один проход и объединяет в единый DataFrame с колонкой ticker. Векторизованная генерация признаков: класс OHLCVFeatureExtractor обрабатывает весь DataFrame целиком через groupby('ticker').apply() для расчёта троичных трендов, преобразование в текст идёт через featuresto_text_vectorized() с np.select() вместо циклов — прирост скорости в десятки раз. Скользящая валидация: WalkForwardValidator обучает модель на окне в 252 дня, тестирует на следующих 21 дне, затем сдвигает окно на 21 день вперёд.

Модель — AutoModelForSequenceClassification (DistilBERT) для бинарной классификации. Токенизатор distilbert-base-uncased, максимальная длина — 128 токенов.

Метрики формируются по каждому тикеру отдельно, на его данных OHLCV (файлы.txt в /Data/Tinkoff). В каждом тикере — несколько временных фолдов (1 год train, 1 месяц test). Потом усредняются по фолдам и по тикерам.

Финальные числа (accuracy, f1, auc) — это усреднение по: все тикеры × все временные окна (folds). В итоге в analyze_results() строится гистограмма AUC и топ-15 тикеров по качеству.

Проблемы и их решение

Изначально паттерны кодировались вымышленными словами («Кибас», «Гапот»), чтобы модель не опиралась на предобученные знания. Но это превращало BERT в обычный классификатор на случайных токенах. Решением стал переход на естественные фразы price rising strongly, near resistance. Модель теперь задействует понимание финансовой терминологии.

Моя RTX 5060 Ti (архитектура Blackwell, SM_120) оказалась слишком новой для стабильных PyTorch. Ошибка «no kernel image available» блокировала вычисления. Решением стал Docker-образ с CUDA 12.8 и nightly-сборкой PyTorch.



В процессе теста 40 бумаг

Обработка каждого файла в цикле была узким местом. Я переработал код для пакетной обработки: все котировки загружаются в единый DataFrame, а признаки генерируются одним векторизованным вызовом. Благодаря этому тесты для более чем 200 акций завершились неожиданно быстро — всего около получаса.

Несовместимость transformers и tokenizers ломала сборку Docker. Зафиксировал работающие версии: transformers==4.35.2, она требует tokenizers==0.15.0. Затем pandas изменил поведение groupby.apply, transformers удалил старые аргументы из API. Адаптация кода и жёсткая фиксация версий в requirements.txt.

В Docker контейнер создавал файлы от root. Переключение на --user "\$(id -u):\$(id -g)" решило проблему с правами, но библиотеки пытались писать кэш в /.cache и падали с PermissionError. Решил что в переменные окружения в `run.sh` перенаправляют кэши в доступные пути: HF_HOME, TRANSFORMERS_CACHE идут в /workspace/.cache, MPLCONFIGDIR — в /tmp.

Детали конфигурации и обучения

Конфигурация признаков: short_window=3, medium_window=7, long_window=14.
Валидация: train_size=252, test_size=21, step_size=21.

Гиперпараметры: learning_rate=2e-5, batch_size=32, epochs=2, weight_decay=0.01, fp16=True. EarlyStoppingCallback(patience=2) останавливает обучение при стагнации лосса.

Оценка — усреднение метрик по 3 фолдам на тикер. Для каждого фолда: accuracy, precision, recall, F1, AUC-ROC. Финальный результат — среднее и стандартное

отклонение AUC.

Выводы

Эксперимент подтвердил: идея семантического кодирования рыночных данных — рабочая и перспективная, но в текущей реализации она не дала статистически значимого результата. Модель действительно различала рыночные ситуации, но слабо — AUC в среднем по полной выборке составил около 0.53. Это слишком мало, чтобы использовать прогнозы в торговле, но достаточно, чтобы признать: языковая модель способна уловить элементарные закономерности, если данные поданы в привычной ей форме — в виде текста.

Главная ценность проекта не в точности предсказаний, а в том, что удалось пройти весь путь от сырого CSV с котировками до работающего ML-конвейера, полностью воспроизводимого в Docker. Каждый этап — от векторизации признаков до скользящей валидации — отлажен и готов к повторным экспериментам. Это не «ещё одна нейросетка для трейдинга», а инженерный прототип, на котором можно проверять новые идеи: другие схемы описания графиков, языковые модели большего масштаба, мультимодальные подходы.

Фактически проект стал мини-лабораторией по исследованию того, как LLM «видит» рынок. И если заменить DistilBERT на современные архитектуры вроде LLaMA или Mistral с дообучением на финансовых текстах, потенциал подхода может проявиться гораздо сильнее.

Повторите мой путь: код на GitHub

Я специально оформил всё так, чтобы любой мог запустить эксперимент у себя. Достаточно скачать архив котировок, собрать контейнер и запустить [run.sh](#) — среда поднимется автоматически с нужными версиями библиотек и CUDA.

Проект открыт:

👉 github.com/empenoso

Если вы хотите повторить эксперимент, улучшить разметку или попробовать другую модель — все инструменты уже готовы. Мой результат — это не финальный ответ, а отправная точка для следующего шага: сделать так, чтобы языковая модель действительно читала графики, а не просто угадывала направление.

Автор: Михаил Шардин

🔗 [Моя онлайн-визитка](#)

📱 [Telegram «Умный Дом Инвестора»](#)

14 октября 2025

— — — — —

🔥 3 😊

💬 5 📄 ➦

**Искусственный интеллект**

5.2K постов • 11.6K подписчиков

[Добавить пост](#)[Подписаться](#)**Правила сообщества**

ВНИМАНИЕ! В сообществе запрещена публикация генеративного контента без детального описания промтов и процесса получения публикуемого результата.

...

[Подробнее](#) [Все комментарии](#)[Автора](#)[Раскрыть 5 комментариев](#)

Чтобы оставить комментарий, необходимо [зарегистрироваться](#) или [войти](#)



[Redacted name]



[Redacted name]

[Redacted comment text]



[Redacted name]



[Redacted name]

[Redacted comment text]



[Redacted name]



[Redacted name]

[Redacted comment text]



