ChatGPT

План решения задач соревнования Secure FinAI Contest 2026

Secure FinAI Contest 2026 включает пять задач, направленных на финансовые LLM и агентские системы: адаптивную оценку FinLLM, надёжный поисковый агент, арбитраж в рыночных предсказаниях, предсказание успеха стартапов (VC) и агентское трейдинг-решение. Ниже приведён поэтапный план решения каждой из задач, включая архитектуру, методы и метрики.

Задача I: Адаптивное тестирование и бенчмаркинг FinLLM

Для оценки финансовых LLM-предложений используются задачи анализа тональности, классификации новостей, NER и вопросно-ответные задачи (ConvFinQA). Например, модель BloombergGPT превосходила другие модели на четырёх из пяти финансовых задач (FiQA-SA, FPB, Headlines, ConvFinQA) и была второй в NER [1]. Обычные бенчмарки фиксированы и выбирают все задачи сразу, тогда как **адаптивный** подход (Item Response Theory) динамически подбирает новые примеры по сложности, сокращая число тестов при сохранении точности оценки [2].

- **Данные и метрики:** Использовать стандартный набор задач FinLLM (FPB – анализ сентимента, FiQA-SA – аспектно-ориентированный сентимент, News Headlines – классификация новостей, NER – распознавание сущностей, ConvFinQA – диалоговый финансовый Q&A). Метриками выступают точность/F1 для классификации и сентимента, точность (exact match) для QA [3] [4].
- **Адаптивное тестирование:** Разбить тестовый набор на уровни сложности (например, по стабильности ответов моделей или по метрикам Hellinger-IRT), как в системе ATLAS [2]. Модель начинает с задач средней сложности и по её успехам автоматически получает более сложные примеры. Это снижает объём испытаний (до ~10%) при сохранении качества измерения [2].
- **Тестовый пайплайн:** Для каждого подзадачи реализовать сценарии: генерировать вопросы, запускать модели и собирать ответы, оценивать с учётом допусков (например, допустимое округление в ConvFinQA). Учесть эффективность вывода (ограничение времени ответа). Сравнивать не только чистую точность, но и способность к точному ответу на числовые вопросы (без «галлюцинаций»).

Задача II: Надёжный поисковый агент FinSearch (FinSearchComp)

Финансовый поисковый агент должен надёжно искать и извлекать данные, не создавая неверных ответов («галлюцинаций»). Задача FinSearchComp специально создана для оценки таких агентов: это первый открытый бенчмарк для реалистичных задач финансового поиска и рассуждения [5]. Набор данных состоит из 635 вопросов, составленных экспертами (по 70+ аналитикам) и делится на три подзадачи: *Time-Sensitive Data Fetching* (реальные текущие данные), *Simple Historical Lookup* (простой поиск исторических цифр) и *Complex Historical Investigation* (сложные вычисления на исторических данных) [5] [6].

- **Архитектура агента:**

- **Модуль поиска:** использовать гибридный подход: комбинировать web-поиск (например, финансовые API, поисковые системы типа SerpAPI или Bing) и локальный поиск по векторной БД документов (например, `ChromaDB` или `Weaviate`) для релевантных финансовых текстов.
- **Модуль извлечения данных:** после нахождения документов или ответов выполнять стринг-парсинг и извлечение чисел/дат. Для повышения точности применить векторные сравнения и числовые методы (разделение цифр на токены, как в BloombergGPT [1], или использование специализированных библиотек вроде `regex` / `tregex`).
- **Модуль верификации и рассуждения:** использует LLM для составления цепочки рассуждений (Chain-of-Thought) с явным указанием шагов. Включает механизм самопроверки: если ответ числовой, перепроверять его другим источником или способом. Если уверенность низкая, возвращать «Не достаточно данных», чтобы избежать домыслов. Такой отказ-«фейловый» режим важен для финансового агента, где ошибка недопустима.
- **Логирование и объяснения:** фиксировать входные запросы, найденные факты/фрагменты, промежуточные шаги рассуждений, итоговый ответ. Это обеспечит проверяемость и даст материал для оценки качества разъяснений.
- **Особое внимание надёжности:** Главная цель – отсутствие галлюцинаций. Решения: использовать строгие правила валидации чисел (например, сравнивать процентные значения в разных форматах как равные [7]), исключать неоднозначные источники. Можно внедрить Multi-hop RAG: если ответ должен включать вычисление (задача 3), агент сначала собирает все нужные данные, затем просит LLM сделать расчёт.
- **Оценка качества:** проверить агент на отложенной части FinSearchComp; измерить процент точных ответов, долю отказов в ответе и частоту отклонённых галлюцинаций. Для анализа обоснованности ответов можно применять «LLM-судью»: попросить GPT-4 оценить качество выданной цепочки рассуждений по метрикам когерентности.

Задача III: Арбитраж на рынках предсказаний (Kalshi vs Polymarket)

Необходимо построить торгового агента, выявляющего арбитражные возможности между двумя Prediction Markets (Kalshi и Polymarket) для бинарных исходов спортивных событий. Данные включают реальные ленты торгов (order book, сделки) и метаданные рынков, а также *Sports Sentiment Feed* (новостные заголовки, обновления о составах, травмах).

- **Настройка данных:** получать историю торгов обоих рынков через WebSocket/API. Сопоставить рынки по исходам (той же игре в Kalshi и Polymarket). Интегрировать sentiment-данные (например, счётчик позитивных/негативных новостей).
- **Стратегия арбитража:** разработать алгоритм, сравнивающий цены на идентичный исход: если цена «Да» в одной системе ниже, чем «Да» (или эквивалентная «Нет») в другой с учётом комиссий, совершать парные сделки (купить дешёвое, продать дорого). Можно начинать с правил: например, при расхождении более порогового уровня. Для прогнозирования вероятных отклонений применять ML/PL-модель, обученную на исторических данных: вектор сигналов может включать относительную разницу цен, книгу ордеров, моментальные объемы и sentiment-показатели.
- **Агент vs RL:** можно реализовать простую стратегию (greedy arbitrage) или обучить RL-агента (Q-обучение/Policy Gradient) в симуляции, где среда – исторические данные обоих

рынков. Поскольку данные временные и детерминированы, RL возможно заменить оптимизацией стратегии (например, генетические алгоритмы).

- **Тестирование (paper trading):** провести backtest на исторических данных, воспроизводя действия агента и вычисляя итоговую прибыль. Считается зафиксированный доход (cumulative return) и риск-метрики (Sharpe, Max Drawdown). При отсутствии реальных денег использовать виртуальный капитал.
- **Метрики и риски:** основной критерий – прибыльность стратегии. Оценивать *Cumulative Return (CR)* и вторичные *Sharpe Ratio, Max Drawdown* и пр. (как указано в задаче V, но переносимо к этой задаче). Учитывать ликвидность: во время симуляции агент не должен совершать сделок, которые он не смог бы выполнить в жизни.

Примечание: Задачи трейдинга активно рассматривались в FinRL Contests – уже предлагались алгоритмические и RL-решения для арбитража и криптоторговли [8]. Можно изучить их стартер-киты (GPU-оптимизированные среды) и примеры стратегий.

Задача IV: AI для венчура – VCBench (прогноз успеха стартапа)

VCBench – первый публичный бенчмарк для задачи прогнозирования успеха стартапа по профилю основателя [9]. Данные содержат **9 000 анонимизированных профилей** основателей, из которых ~810 помечены как «Success» (либо вышли на IPO/слияние > $500M/получили топ-фондирование) [9]. Необходимо классифицировать каждый профиль как успешный или нет (метрика – F1-score).

- **Предварительная обработка:** профили представлены в стандартизированном текстовом формате (его формат указан в документации VCBench [9]). Тексты уже анонимизированы, чтобы исключить утечку личности. Убрать недостающие поля и шумовые атрибуты, нормализовать описания (унификация названий должностей и степеней).
- **Моделирование:** использовать LLM (например, GPT-4o или Claude 3) либо fine-tune больший модельный LLM на эту задачу. Подготовить подсказки (prompts) для классификации: включить описание профиля и запросить label. Можно также обучить lightweight классификатор (Transformers для text classification) на репрезентативных примерах.
- **Учёт дисбаланса:** разметка несбалансирована (~9% успехов). Применять технику балансировки (взвешивание классов, oversampling успешных профилей).
- **Валидация и метрики:** разбить данные на train/val/test. Оценивать модель по F1-score (задача требует оптимизации именно этой метрики). По желанию отобразить Precision/Recall для понимания ошибок.
- **Интерпретируемость:** анализ причин правильно/неправильно классифицированных профилей (например, с помощью LLM-секвенса внимания). Это покажет, какие признаки модель считает важными (коррелирует с «что видит инвестор»).

Задача V: Агентский трейдинг с обоснованием

Задача требует принять ежедневное торговое решение (Buy/Hold/Sell) для двух активов (ETH-крипта и акции MSFT) на основе данных в каждый день, а также предоставить короткое текстовое обоснование ($\leq$50 слов), опирающееся на имеющиеся сигналы (цены, новости, фундамент).

Оценка – прибыльность и риск метрик (Cumulative Return, Sharpe, Max Drawdown) в условиях реального временного прогресса.

- **Данные и сигналы:** каждая дата снабжена историческими данными цен до текущего дня, синтезированным *newsflow* (заголовки/новости), метками тренда (momentum) и фундаментальной информацией. Задача – предсказать рыночное действие без утечки будущей информации.
- **Модель агента:** можно использовать LLM в режиме вопрос-ответа: сформировать промпт, который содержит последние данные (например, 30-дневные цены, краткое резюме новостей), и попросить LLM выдать «решение + обоснование». Либо реализовать двухэтапную систему: ML/RL-алгоритм (XGBoost, небольшая нейронная сеть или RL-policy) генерирует действие, а LLM формулирует пояснение, используя факты из входных данных.
- **Обоснование (рационал):** требование обоснования повышает интерпретируемость. Нужно обеспечить, чтобы текст был «evidence-grounded»: указывать конкретные данные, например “Buy, because последние данные Ethereum показывают восходящий тренд из новостей...” и т.п.
- **Оценка и протокол:** расчёт CR, Sharpe, MaxDrawdown и волатильности по итогам всей тестовой серии. Важно соблюдать хроно-логию: решения подаются ежедневно в порядке дат, финальный отчёт можно сдать либо сами ежедневные даты-действия, либо код, который их генерирует.
- **MLOps:** для устойчивости хранить версии моделей, фиксировать seed и дату генерации. Это соответствует «leakage-resistant» протоколу.

  Примечание: В FinRL тоже рассматриваются агентские торговые задачи с учётом риска и интерпретации решений [8]. В такой комплексной задаче важна инфраструктура (автоматический бэктестинг, учёт транзакционных издержек и др.).

Общие рекомендации по реализации и MLOps

- **Команда и инструменты:** оптимально 2–4 человека с разными навыками (ML-архитектура, бэкенд, DevOps). Завести общий Git-репозиторий (GitHub/GitLab) и канбан-доску (GitHub Projects или Jira), отражающую разбивку на эпики (Data, Модели, Инфраструктура, Документация).
- **Архитектура:** применять микросервисный подход (например, отдельный сервис для RAG-поиска, сервис для LLM-рассуждений), оборачивать модели в API (FastAPI). Хранить эмбеддинги/данные в базе (ChromaDB, Pinecone). Использовать `pydantic` для валидации данных.
- **RAG-пайплайн:** при поиске и извлечении информации использовать `LangChain` или `LlamaIndex`. Формировать запросы к LLM с явными инструкциями (инструкционные подсказки). По возможности применять переранкинг (reranking) найденных документов с помощью модели.
- **Модели:** выбирать баланс качества/скорости. Можно использовать закрытые модели (GPT-4o, Claude) или открытые (Qwen2.5-72B, Llama с дообучением). Для вычислений можно обращаться к Python (например, через `tools` в LangChain) для точности в Task II/III.
- **Логирование и трассировка:** логировать входы/выходы каждого шага агента. Можно хранить трассировки в W&B или ElasticSearch для анализа. Это продемонстрирует «верифицируемость» вычислений.
- **Мониторинг качества:** настроить метрики (точность на тестовых вопросах, финансовые метрики), мониторить их в процессе обучения. Оценить воздействие обновлений моделей через A/B-тесты на отложенной выборке (насколько изменилась точность или прибыльность).

- **Безопасность и приватность:** использовать только публичные данные, не хранить личную информацию. Если доводится работать с приватными источниками (не описано в задачах), то подключать методы приватной обработки (дифференциальная приватность или TEE).

Итоги и артефакты

По окончании рекомендуется подготовить: реализованный код с `README` и инструкциями, технический отчёт (описание архитектуры, экспериментов, метрик), презентацию по каждому ключевому моменту (архитектура агента, борьба с галлюцинациями, управление рисками) и демонстрационный скрипт. В отчёте следует обосновать выбор решений (например, почему выбран именно RAG-подход для Task II) с опорой на результаты на наборах данных [5] [9]. Такой всесторонний документ станет наглядным доказательством решения всех задач конкурса.

---

[1] BloombergGPT: A Large Language Model for Finance
https://arxiv.org/html/2303.17564v3

[2] Adaptive Testing for LLM Evaluation: A Psychometric Alternative to Static Benchmarks | OpenReview
https://openreview.net/forum?id=iX39LhePyd

[3] [4] Exploring Large Language Models for Financial Applications: Techniques, Performance, and Challenges with FinMA
https://arxiv.org/html/2510.05151v1

[5] [6] Paper page - FinSearchComp: Towards a Realistic, Expert-Level Evaluation of Financial Search and Reasoning
https://huggingface.co/papers/2509.13160

[7] ByteSeedXpert/FinSearchComp · Datasets at Hugging Face
https://huggingface.co/datasets/ByteSeedXpert/FinSearchComp

[8] [2504.02281] FinRL Contests: Benchmarking Data-driven Financial Reinforcement Learning Agents
https://arxiv.org/abs/2504.02281

[9] VCBench: Benchmarking LLMs in Venture Capital
https://arxiv.org/html/2509.14448v1