Супер-интеллект ансамбль для поиска

OpenRouter Fusion MCP для Codex

Что это?

Представь, что у тебя есть вопрос, где ошибка стоит дорого: стратегия, продукт, рынок, архитектура, исследование, юридический риск, важная покупка, спорная гипотеза. Обычный чат с ИИ может ответить быстро и уверенно, но уверенность не равна правде.

Решение: консенсус моделей. Вместо того чтобы доверять одному ответу одной модели, ты запускаешь совет из нескольких ИИ-экспертов. Они независимо разбирают задачу, спорят с идеей с разных сторон, подсвечивают риски и помогают получить более сильный итоговый вывод.

Это рабочий способ получить второй, третий и пятый взгляд на сложный вопрос за один запуск.

Суть: локальный MCP-сервер добавляет в Codex инструмент `fusion`. Когда ты просишь Codex запустить Fusion, запрос отправляется в OpenRouter. Несколько моделей анализируют задачу параллельно, а модель-судья собирает результат: консенсус, противоречия, частичное покрытие, уникальные идеи, слепые пятна и финальный ответ.

Fusion особенно полезен, когда тебе нужен не просто "ответ", а проверенная позиция: что вероятнее, где риски, в чём модели не согласны и что стоит перепроверить вручную.

Для кого

- **Маркетинг:** проверить оффер, сегмент, УТП, позиционирование, креативы, лендинг, рекламную гипотезу.
  Пример: `Запусти fusion и разбери, почему этот оффер для B2B SaaS может не конвертить. Дай 5 гипотез, риски и варианты тестов.`
- **Бизнес и стратегия:** сравнить варианты выхода на рынок, найти слабые места в стратегии, оценить конкурентов, подготовиться к переговорам.
  Пример: `Запусти fusion и сравни 3 стратегии монетизации для моего продукта: подписка, usage-based и enterprise deals.`
- **Основатели и руководители:** принять решение, где важно увидеть не только плюсы, но и скрытые риски.
  Пример: `Запусти fusion и выступи как совет директоров: стоит ли нам делать pivot в сторону enterprise-клиентов?`
- **Продукт и UX:** оценить фичу, CJM, onboarding, механику удержания, pricing page или пользовательский сценарий.
  Пример: `Запусти fusion и найди слабые места в onboarding для нового пользователя. Дай рекомендации по приоритетам.`

- **Разработка и архитектура:** сравнить технические подходы, найти edge cases, риски миграции, слабые места в API/биллинге/инфраструктуре.
  Пример: `Запусти fusion и сравни варианты хранения истории чатов: SQLite, Postgres и event log. Дай рекомендацию для MVP и для роста.`
- **Исследования, аналитика и R&D:** собрать несколько независимых рассуждений, найти противоречия, сформировать план проверки.
  Пример: `Запусти fusion и составь карту аргументов за и против этой научной гипотезы. Отдельно укажи, что нужно проверять источниками.`
- **Обучение и разбор сложных тем:** получить объяснение с разных сторон и увидеть, где есть спорные места.
  Пример: `Запусти fusion и объясни reinforcement learning с нуля, но отдельно покажи разные интуиции и типичные ошибки новичков.`
- **Личные решения с высокой ценой ошибки:** сравнить варианты покупки, обучения, переезда, выбора подрядчика или инструмента.
  Пример: `Запусти fusion и помоги выбрать между двумя ноутбуками для разработки и локальных моделей. Сравни по рискам, цене и сроку службы.`

Важно: консенсус моделей не гарантирует истину. Он снижает риск одиночной ошибки и помогает увидеть больше углов, но факты, цифры, законы, медицину, финансы и критичные решения всё равно нужно проверять по источникам.

Пример состава моделей

Например, можно собрать консенсус из нескольких моделей:

```
moonshotai/kimi-k2.7-code
qwen/qwen3.7-plus
minimax/minimax-m3
qwen/qwen3.7-max
x-ai/grok-4.3
```

А моделью-судьёй поставить:

```
deepseek/deepseek-v4-pro
```

Такой ансамбль часто даёт более устойчивый и разносторонний анализ, чем одиночная модель: одна модель сильнее в коде, другая в рассуждении, третья лучше видит продуктовые риски, четвёртая жёстче критикует. На выходе ты получаешь не один голос, а сводку нескольких позиций.

Но качество зависит от доступности моделей в OpenRouter, формулировки запроса, лимитов, стоимости и текущего состояния провайдеров.

Что понадобится

1. Установленный Codex CLI.
2. Файл локального MCP-сервера:

   ```
   tools/openrouter_fusion_mcp.py
   ```

3. Аккаунт OpenRouter:

   ```
   https://openrouter.ai
   ```

4. API-ключ OpenRouter:

   ```
   https://openrouter.ai/workspaces/default/keys
   ```

5. Файл конфигурации Codex:

   ```
   ~/.codex/config.toml
   ```

Как это работает

После настройки Codex запускает локальный Python-файл как MCP-сервер:

```
/opt/homebrew/bin/python3
/Users/artem/Desktop/chatbot/tools/openrouter_fusion_mcp.py (пути показаны как
пример!)
```

Дальше происходит так:

```
Codex
  -> запускает локальный MCP server
  -> спрашивает список tools
  -> видит tool fusion
  -> вызывает fusion с твоим вопросом
  -> MCP server отправляет запрос в OpenRouter
  -> OpenRouter запускает openrouter:fusion
  -> ответ возвращается обратно в Codex
```

Файл `openrouter_fusion_mcp.py` не является обычной CLI-программой. Если просто запустить его руками, он будет молчать, потому что ждёт MCP/JSON-RPC сообщения через `stdin`.

Установка

1. Положи файл в проект

В корне проекта должна быть папка `tools`:

```
tools/openrouter_fusion_mcp.py
```

Если папки нет:

```
mkdir -p tools
```

Скопируй файл `openrouter_fusion_mcp.py` внутрь этой папки.

2. Узнай путь к Python

В терминале:

```
which python3
```

На Mac с Homebrew часто будет:

```
/opt/homebrew/bin/python3
```

На других системах путь может быть другим, например:

```
/usr/bin/python3
```

В config нужно вставлять именно свой путь.

3. Узнай абсолютный путь к проекту

Перейди в проект и выполни:

```
pwd
```

Пример:

```
/Users/artem/Desktop/chatbot
```

Тогда путь к MCP-файлу будет:

```
/Users/artem/Desktop/chatbot/tools/openrouter_fusion_mcp.py
```

Настройка config.toml

Открой глобальный config Codex:

```
nano ~/.codex/config.toml
```

`sudo` обычно не нужен. Если файл не открывается из-за прав, сначала проверь, что ты редактируешь именно свой `~/.codex/config.toml`.

В конец файла добавь блок:

```
[mcp_servers.openrouter_fusion]
command = "/opt/homebrew/bin/python3"
args = ["/Users/artem/Desktop/chatbot/tools/openrouter_fusion_mcp.py"]
cwd = "/Users/artem/Desktop/chatbot"
startup_timeout_sec = 30
tool_timeout_sec = 400
enabled_tools = ["fusion"]
default_tools_approval_mode = "prompt"

[mcp_servers.openrouter_fusion.env]
OPENROUTER_API_KEY = "sk-or-v1-..."
```

Что заменить:

- `command` — путь из команды `which python3`.
- `args` — абсолютный путь к `tools/openrouter_fusion_mcp.py`.
- `cwd` — абсолютный путь к проекту.
- `OPENROUTER_API_KEY` — твой ключ OpenRouter.

Сохранить в `nano`:

```
Ctrl + O
Enter
Ctrl + X
```

Можно открыть файл и любым обычным редактором, если так удобнее.

Важное про API-ключ

Для первой проверки можно вставить ключ прямо в `config.toml`.

Но если конфиг лежит внутри проекта, не коммить его в git. API-ключ нельзя публиковать.

Более безопасный вариант — держать ключ в переменной окружения, но для быстрого старта проще использовать блок:

```
[mcp_servers.openrouter_fusion.env]
OPENROUTER_API_KEY = "sk-or-v1-..."
```

Перезапусти Codex

После изменения `config.toml` полностью закрой и заново открой Codex CLI.

Codex читает MCP-конфиг на старте. Если не перезапустить Codex, новый инструмент может не появиться.

Как проверить, что Codex видит MCP server

В терминале:

```
codex mcp list
codex mcp get openrouter_fusion
```

В выводе должно быть примерно:

```
openrouter_fusion
  enabled: true
  transport: stdio
  command: /opt/homebrew/bin/python3
  args: /.../tools/openrouter_fusion_mcp.py
  enabled_tools: fusion
```

Если путь в `args` неправильный, сервер не стартует.

Как запустить Fusion из Codex

Пиши обычным языком, но явно упоминай `fusion`, чтобы Codex понял, что нужно вызвать MCP-инструмент.

Минимальный запрос:

```
Запусти fusion и сравни подходы к реализации billing в этом проекте.
```

Для бизнес-гипотезы:

```
Запусти fusion и проверь гипотезу: "малому бизнесу нужен AI-ассистент для
обработки входящих заявок". Разбери спрос, боли, возражения, конкурентов, каналы
продаж и первые тесты.
```

Для маркетинга:

```
Запусти fusion и разбери этот лендинг как performance marketer, продуктовый
маркетолог и скептичный покупатель. Найди слабые места, предложи новые офферы и
A/B тесты.
```

Для разработки:

```
Запусти fusion и сравни два технических подхода: хранить billing ledger в SQLite
append-only файлах или сразу перейти на Postgres. Укажи риски, сложность миграции
и рекомендацию для MVP.
```

Для решения:

```
Запусти fusion и помоги принять решение: нанимать senior-разработчика сейчас или
закрыть задачу подрядчиком. Дай аргументы за/против, риски, условия выбора и
итоговую рекомендацию.
```

С ручным выбором моделей:

```
Вызови fusion с analysis_models=["moonshotai/kimi-k2.7-code", "qwen/qwen3.7-max"]
и judge_model="deepseek/deepseek-v4-pro".
```

Хорошая формула запроса:

```
Запусти fusion и [что нужно решить].
Контекст: [коротко кто мы, что делаем, ограничения].
Нужно получить: [формат ответа].
Оцени: [риски, альтернативы, стоимость, сроки, спорные места].
В конце дай: [итоговую рекомендацию и что проверить вручную].
```

Что можно просить передавать Codex как доп аргументы

Основные аргументы который поддерживает инструмент :

```
prompt
model
system_prompt
analysis_models
judge_model
max_tool_calls
max_completion_tokens
temperature
reasoning
force
include_raw
```

Минимально нужен только:

```
prompt – как раз тот вопрос, который мы задаем (его составляет агент)
```

Остальное можно не указывать: сервер подставит значения по умолчанию.

Диагностика

Если сервер не стартует, временно добавь в config:

```
[mcp_servers.openrouter_fusion.env]
OPENROUTER_API_KEY = "sk-or-v1-..."
OPENROUTER_FUSION_MCP_DEBUG = "1"
OPENROUTER_FUSION_MCP_LOG = "/private/tmp/openrouter_fusion_mcp.log"
```

После перезапуска Codex проверь лог:

```
sed -n '1,160p' /private/tmp/openrouter_fusion_mcp.log
```

Нормальный старт выглядит так:

```
[openrouter-fusion] server started
[openrouter-fusion] received method='initialize'
[openrouter-fusion] received method='notifications/initialized'
[openrouter-fusion] received method='tools/list'
```

Если ты видишь:

```
[openrouter-fusion] received method='tools/call'
[openrouter-fusion] calling OpenRouter ...
[openrouter-fusion] OpenRouter response received status=200
```

значит запрос реально ушёл в OpenRouter и вернулся.

Частые ошибки

MCP startup timed out

Что значит: Codex запустил процесс, но не получил нормальный MCP handshake.

Что проверить:

- правильный ли путь в `args`;
- существует ли файл `tools/openrouter_fusion_mcp.py`;
- нет ли синтаксической ошибки в Python-файле;
- перезапустил ли ты Codex после изменения config.

Проверка синтаксиса:

```
python3 -m py_compile tools/openrouter_fusion_mcp.py
```

connection closed: initialize response

Чаще всего Python-файл упал при старте.

Проверь:

```
python3 -m py_compile tools/openrouter_fusion_mcp.py
```

Если есть синтаксическая ошибка, Codex не сможет запустить MCP-сервер.

В логе есть только server started

Значит процесс стартовал, но Codex не дошёл до handshake или transport не совпал.

Текущий файл поддерживает:

```
JSONL MCP transport
Content-Length framing
```

OpenRouter request failed: nodename nor servname provided

Это DNS/network ошибка. Обычно значит, что среда, где запущен процесс, не имеет доступа к сети.

Проверь интернет, VPN, прокси, sandbox-настройки и доступность `openrouter.ai`.

Запрос висит слишком долго

Fusion может работать долго, потому что запускает несколько моделей.

Что можно сделать:

- уменьшить `analysis_models`;
- поставить `max_tool_calls = 1`;
- поставить `max_completion_tokens = 512`;
- использовать более быструю outer model;
- увеличить `tool_timeout_sec`, если реально нужен длинный анализ.

Практические советы

- **Не запускай Fusion на мелочь.** Для вопроса "как назвать переменную" это лишнее. Для стратегии, архитектуры, конкурентов, pricing, рисков, продуктовой гипотезы — самое то.
- **Давай контекст.** Плохой запрос: `Оцени идею`. Хороший запрос: `Оцени идею AI-ассистента для стоматологий в России. ЦА: клиники 3-20 врачей. Цель: понять, стоит ли делать MVP.`

- **Проси спор, а не только ответ.** Хорошая формулировка: `Покажи, где модели могут не согласиться, какие есть аргументы против и какие факты надо проверить.`
- **Проси структуру.** Например: `Дай ответ в формате: краткий вывод, аргументы за, аргументы против, риски, что проверить, рекомендация.`
- **Ограничивай масштаб.** Если задача большая, проси сначала стратегический анализ, потом отдельным запуском — план действий. Один огромный запрос может быть дороже, медленнее и менее точным.
- **Следи за стоимостью.** Чем больше моделей в `analysis_models`, чем длиннее контекст и чем выше `max_completion_tokens`, тем дороже и дольше запрос.
- **Для быстрых проверок ставь лимиты.** Например: `max_tool_calls=1`, `max_completion_tokens=512`. Для глубокого анализа можно увеличивать.
- **Не доверяй консенсусу слепо.** Если все модели уверенно ошиблись из-за неверного исходного факта, консенсус тоже будет неправильным. Критичные факты проверяй отдельно.
- **Если OpenRouter ругается на модель, замени её.** Иногда конкретная модель временно недоступна у провайдера. Убери её из `analysis_models` или выбери более стабильную.
- **Если запрос висит долго, уменьши состав ансамбля.** Начни с 2-3 моделей, потом расширяй, если ответ реально нужен глубже.
- **Для кода и архитектуры проси edge cases.** Например: `Отдельно найди проблемы с миграцией, конкурентным доступом, idempotency, billing и логированием.`
- **Для бизнеса проси go/no-go.** Например: `В конце дай решение: делать, не делать или проверить через быстрый тест.`