Чек-лист настройки конвейера WhisperX

Пословные таймкоды + диаризация говорящих на своей инфраструктуре

1 · Четырёхстадийный конвейер

#	Стадия	Модель	Что вы получаете
1	Детекция речевой активности	pyannote VAD	Найти речь; Cut & Merge в куски ~30 с по тишине
2	Расшифровка (батчем)	Whisper / faster-whisper	Что сказано; до 70x реального времени
3	Forced alignment	wav2vec2	Когда сказано каждое слово (~50 мс)
4	Диаризация	pyannote.audio	Кто сказал — метки Говорящий A / B

2 · Self-host WhisperX против облачного API

Критерий	WhisperX (self-host)	Облачный API
Стоимость (batch)	Только время GPU (софт бесплатен)	Плата за час, настроено вендором
Пословные таймкоды	~50 мс через forced alignment	Встроены
Диаризация	pyannote, бесплатно, нужен HF-токен	Встроена, один флаг
Где данные	Остаются на ваших серверах	Уходят в облако вендора
Стриминг	Нет — только batch	Да
Когда выбирать	On-prem; ровный высокий объём	Быстро запуститься; рваный объём; стриминг

3 · Чек-лист перед запуском

- ☐ Подтвердили, что вам реально нужны таймкоды слов или метки говорящих — иначе обычный Whisper / faster-whisper проще.
- ☐ Завели токен Hugging Face, приняли условия закрытой модели pyannote и подключили флаг --diarize.
- ☐ Обработали откат выравнивания: выровненные слова несут плотное время; числа/символы наследуют время сегмента — пометьте их как приблизительные.
- ☐ Протестировали тайминг на аудио, полном чисел, цен и имён собственных, а не только на чистой прозе.
- ☐ Почистили шумный вход — выравнивание wav2vec2 деградирует быстрее расшифровки на шумном аудио.
- ☐ Проверили место хранения данных: может ли аудио законно покинуть ваши серверы или обязано остаться on-prem?
- ☐ Запланировали проверку человеком для диаризации там, где ошибка в метках говорящих важна (медицина, юристы).
- ☐ Смоделировали стоимость на реальном batch-объёме (GPU-часы = часы аудио / коэффициент реального времени) и сравнили с облачным API.
- ☐ Нагрузили GPU-конвейер на ожидаемом пике; убедились, что токен не истечёт тихо в проде.