Кто Я?

LinkedIn

СЕЙЧАС:
последние 6 лет - Директор по развитию бизнеса

РАНЬШЕ:
3 года - Product Manager по ML продуктам в oneFactor/МегаФон

ЕЩЕ РАНЬШЕ:
8 лет в Microsoft

ПОЧТИ В ПРОШЛОЙ ЖИЗНИ:
5 лет C++ разработчик

ОБРАЗОВАНИЕ:
МИФИ, Antwerp Management School (MBA)

Какой был подход? – мои размышления

- **В рекомендациях обычно разделяют retriever и ranker модели.**
  В test уже выбраны кандидаты -> тут ranker

- **В ranker хорошо работают деревья**
  Нужен фичеинжениринг - 😫, хочется чтобы «оно само за меня все сделало»
  Им нужно много памяти 🧠 – чего не было
  Значим берем нейронные сети, там памяти нужно только на батч, а фичи «сами как-нибудь»
  Надо было объединяться с кем-нибудь с деревьями

- **Важность очередности?**
  Предположим, что реакция одинаковая при любой очередности –> RNN не будем смотреть

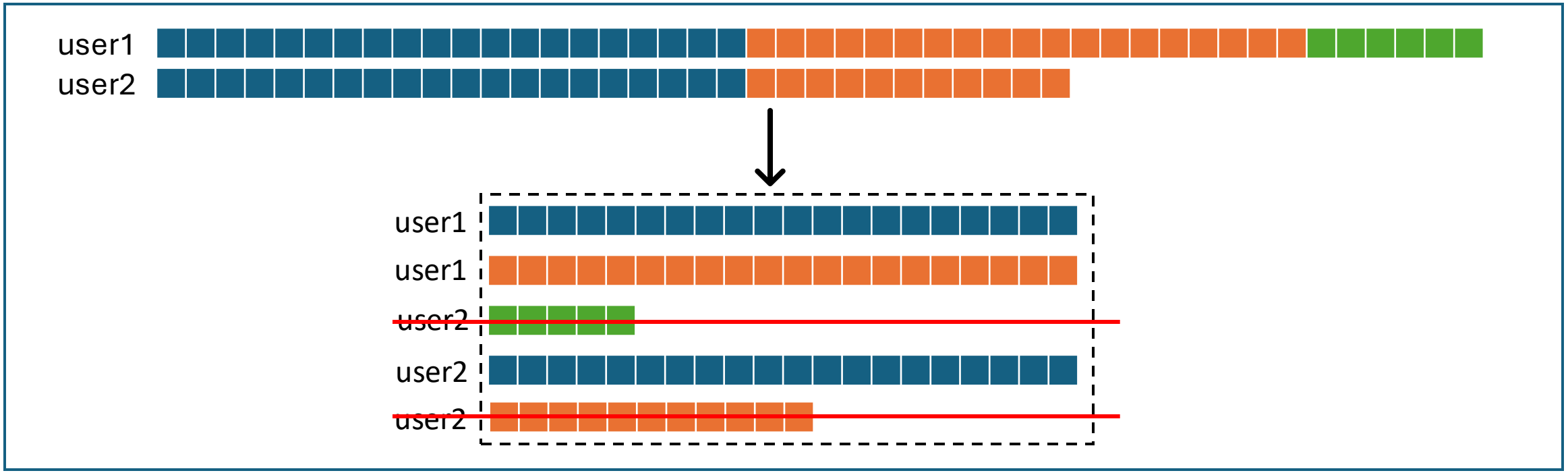
- **У пользователя от 2 до 4566 просмотров видео**:
  Много для трансформеров в лоб –> Нужно резать как-то + все равно нужно будет много GPU
  Надо было попробовать

- **Попробуем что-нибудь простое**
  Засунем все в embedding, прикрутим головы на все и прогоним крупными батчами,
  а дальше будем читать теорию и пробовать легкий тюнинг
  Слишком рано попадаем в топ, много что читаем, но почти ничего не пробуем

Как подавались данные?

Будет учить как в тесте – 1 пользователь и 20 кандидатов

1. Перемешиваем
2. Для каждого пользователя берем все элементы группами по 20
3. Все группы, где нет 20 элементов или все элементы с одинаковым таргетом – выкидываем
4. Перемешиваем
5. batch_size = 4096



Как выглядела модель?

Пользователь

- user_id – embedding (256+128)
- gender - вычитаем 1 (0 или 1)
- age – нормируем

⊕

Конкатенация

Видео

- item_id– embedding (256)
- source_id - embedding (128)
- duration - нормируем
- embeddings – берем как есть

Линейные слои с активацией, нормализацией и dropout

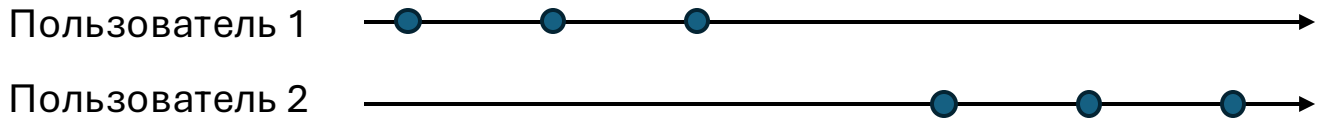
Головы на все что знаем: timespent, like, dislike, share, bookmarks, target

Как выглядел loss?

- like, dislike, share, bookmarks -> nn.BCEWithLogitsLoss()
  возможно надо было поиграть с весами target
- timespent - nn.MSELoss()
- target (-1, 0, 1) – margin loss между элементами с разными таргетами пользователя внутри batch

target	-1	0	1
-1	-	1	2
0	1	-	1
1	2	1	-

Margin между target



Абсолютное значение target может находиться
в разных местах для разных пользователей
оптимизируем разницу

- Все loss просто складывались
  возможно надо было поиграть с весами между разными loss

Как обучалась модель?

- **Первые попытки**
  - С рандомной инициализацией embedding модель сходилась не очень стабильно
  - без GPU за ~30 минут / на GPU (colab/Kaggle) за 10 минут
    public lb: ~0.62 - 0.645

- **Инициализация embedding**
  - Embedding user_id, item_id и source_id инициализируем из ALS
  - ALS на целевое событие, время просмотра и т.д. работал плохо
  - ALS на факт наличия связи (без дополнительных весов) – лучшие результаты
  - Нормализации + ALS за 2 эпохи на CPU за ~30-50 минут / на GPU (colab/Kaggle) за 15 минут -
    public lb: ~0.645 - 0.654

- **Заморозка**
  - Переобучение после двух эпох не получилось остановить –> морозим embedding + дообучаем embedding на одной эпохе (дальше они разваливались = переобучались)
  - 5-8 epoch с замороженными embedding + 1 epoch с размороженными 0.67 – считались около часа на GPU из colab/Kaggle
    public lb: 0.67+

Какие были результаты?

Финальная версия:

- Одиночная модель
  public lb: 0.6731627104511342
- Усреднение этой модели с разными seed для выделения валидационной выборки
  public lb: 0.6773073113118988
- Усреднение с несколькими версиями предыдущих версий (минорные изменения)
  public lb: 0.6781682484518043

СПАСИБО!
ВОПРОСЫ?