ZAKIPKI.HACK

29150_Sergeif

01 Основные идеи

Все просто!

02 Инструменты

Open Source!

03 Достоинства и недостатки

Все преодолимо!

04 Код

Python, ~1024 **строки**.

05 Производительность

Один час, чтобы предсказать их все.

06 Идеи

Что пробовал и куда идти дальше.

Основные идеи

- Решать задачу, как задачу рекомендации новых товаров покупателем (рекомендательные системы).
- LightFM + warp loss!
- Использовать флаг победы с весом числа известных участников как меру заинтересованности поставщика в лоте.
- Векторизовать закупки и поставщиков через эмбединги, тренировать нейронную сеть на парах закупка-поставщик или закупка-закупка. Metric learning (contrastive loss, cosine loss).
- Обогатить данные lightfm эмбедингами пользователей и поставщиков из нейронной сети.
- Месяц, есть ли описание лота - дополнительные признаки.
- Текстовые данные - наименование + описание. TFIdf + SVD.
- Валидация - 2020 год.

LIGHTFM

Быстрая и удобная библиотека для построения векторных представлений пар покупатель-товар или закупка-поставщик в нашем случае. Гибкая, позволяет добавить любые признаки как поставщика, так и закупки.

PYTORCH

Распространенная и хорошо поддерживаемая библиотека построения нейронных сетей. Нейронные сети выглядят хорошим способом обработки текста.

Достоинства и недостатки

Скорость

Подготовка данных и обучение занимают около часа времени на среднем сервере. Возможно обучение на кластере.

Точность

На валидации теоретический максимум 55% полноты, из которых мы получаем 28%, что выглядит не слишком плохо для прототипа..

Обучение

При добавлении новых поставщиков для более точной рекомендации стоит переобучать всю модель, но можно этого не делать с 5-6% потерей точности!

Инструменты

Библиотеки хорошо поддерживаются сообществом, у них миллионы пользователей по всему миру и доступны для любого коммерческого использования.

Код

```
import pandas as pd
import joblib
import numpy as np

train = pd.read_csv('train_data.csv', sep=';', parse_dates = ['min_publish_date'])
test = pd.read_csv('test_data.csv', sep=';', parse_dates = ['min_publish_date'])
labels = pd.read_csv('train_labels.csv', sep=';')
word2inx_map = joblib.load('word2inx_map.joblib')

cat_cols = ['fz','region_code','okpd2_code','additional_code','month','has_lot']
col_cols = ['lot_price']

def process_df(train):
    train['month'] = train['min_publish_date'].dt.month
    train['okpd2_code'] = train['okpd2_code'].fillna('null_okpd2_code')
    train['additional_code'] = train['additional_code'].fillna('null_additional_code')
    train['lot_price'] = np.log1p(train['lot_price']) / 25.0
    train['region_code'] = train['region_code'].astype(str)
    train['has_lot'] = 'lot_'+train.lot_name.isnull().astype(str)
    return train

train = process_df(train)
test = process_df(test)
full_df = pd.concat([train,test], ignore_index=True).reset_index(drop=True)
```

```
interactions = dataset.build_interactions([(row["participant_inn_kpp_anon"],
                                            row["pn_lot_anon"],
                                            1.0 + row['is_winner'] * train_item_participations[row["pn_lot_anon"]])
                                           for index, row in train_labels_df.iterrows()])
```

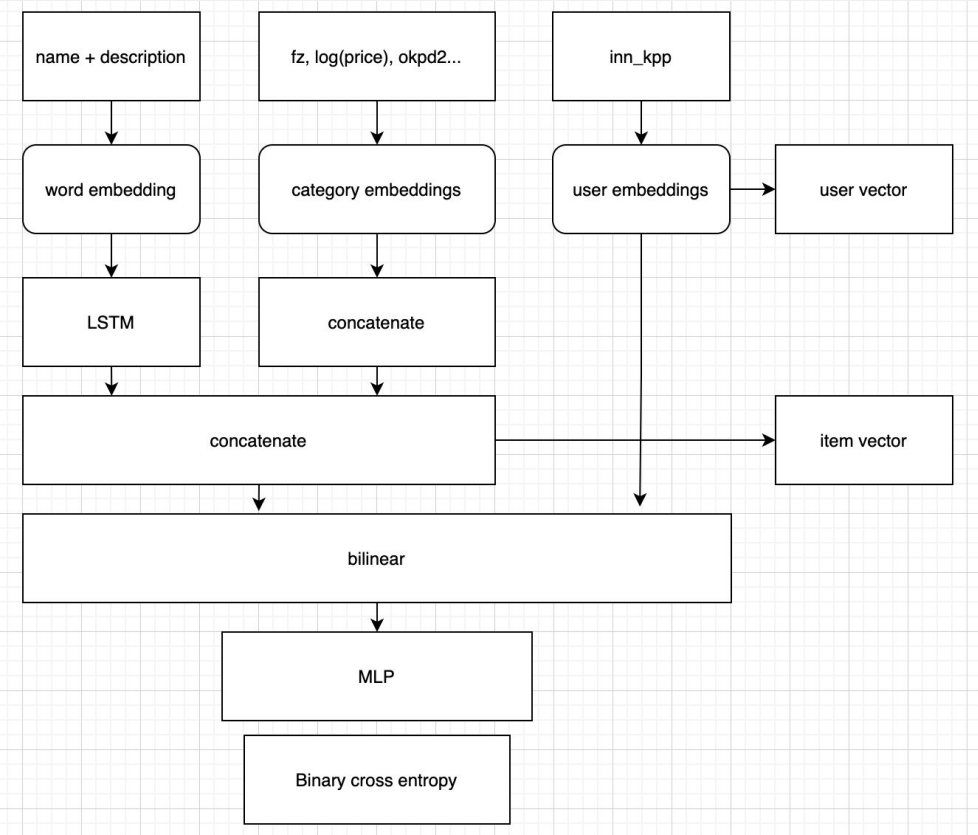
```
user_mappings1, _, item_mappings1, _ = dataset.mapping()
item_mappings = {k: d for d,k in item_mappings1.items()}
user_mappings = {k: d for d,k in user_mappings1.items()}
```

```
inn = user_mappings[0]
valid_interactions = dataset.build_interactions([(inn, x) for x in valid_labels_df.pn_lot_anon.unique()])
valid_users, valid_items = valid_interactions[0].nonzero()
```

```
import tqdm
from scipy.special import softmax

for num_epoch in [100]:
    model = LightFM(loss='warp', no_components=256, random_state=239)
    model.fit(interactions=interactions[0],
              item_features=train_item_features,
              user_features=train_user_features,
              sample_weight=interactions[1],
              epochs=num_epoch, num_threads=8, verbose=True)
```

Нейронная сеть



1 x 1080ti, 100 эпох - 2 часа.

Генерация n positive и столько же случайных negative для каждого лота.

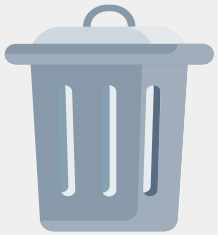
Семплер из metric learning, на каждой эпохе выбираются hard negative из прошлой.

Скорость

На компьютере **8** ядер,
32 гигабайта памяти,
время обучения модели на полном датасете **1** час.

Как улучшить

- Обучить лингвистические модели на специфичном корпусе описаний и получить вектора текста для lightfm.
- Тематическое моделирование (elmo/ulmfit + hdbscan).
- Использовать кластеризацию на полученных векторах и номер кластера как категорию в lightfm.



Еще попробовал

- Metric learning для item-item contrastive, arcface, margin loss. Порядка 0.18 на валидации.
- Предсказывать вектор эмбединга следующей закупки по закупкам пользователя. Очень ресурсоемко, не удалось.
- Тензорное разложение, ALS. Очень сложно закодировать в короткие сроки, оставил.

Буду рад ответить на вопросы!

Спасибо за внимание!