【推荐算法】deepwalk 原理，实战以及工程化

作者：jditlee
原文链接：https://ld246.com/article/1678863113447
来源网站：链滴

【推荐算法】deepwalk原理，实战以及工程化

最近aigc很火啊，chatgpt都能帮忙写文章了，博客里面也发了一篇gpt写的机器学习的简单入门文章ML】机器学习综述 - Neptune (laobiao.fun),感觉很生硬，没有特色，也尝试了别的prompt生成一个性化的文章，还是太过模板化了，工作周报还行，要来写自己的东西还是不能用它。

然后的话，那就开始写自己的东西吧，又是好久没写文字了。

接下来开始我的第一篇推荐算法文章，其实推荐算法最重要的是怎么工程化落地，然而很多推荐领域博客都是讲原理，实战代码的都很少，更不用说讲工程化落地的了，然后我就想写一系列文章来讲一推荐算法的经典模型已经如何工程化，用黑话来说就是怎么让算法持续为商业赋能。我的文章呢采用叙的方式，先说重点结论，然后再拆解里面的知识点，我喜欢这样子来学习，不然一开始一堆知识点能就让我看睡着了。

一、deepwalk原理

直接说精华部分，

就是构造用户-物品（user-item）关系图，然后在图中随机游走构建序列，然后训练item2vec模型，到每一个user和item的embedding

接下来就是具体的知识点了，不想看就直接看实战部分就行了

1.图的概念

用废话文学来说，图简单点说就是图(狗头），就是由**节点**和**边**连结起来的网状结构，其实叫网是不是容易理解一点，节点就是我们的item，可以是用户，物品，以及任何实体，边就是实体之间的关系，如用户a点击了视频b，那么在图里面就是两个节点（用户a，视频b）一条边（点击），边可以是有的和无向的，有向的就叫**有向图**，无向的就叫**无向图**。

2.随机游走

随机游走就是基于已构建的图，遍历每一个节点，随机获取下一个相邻节点，构成item2vec需要的item序列。

好像一句话就说完了，再给一张论文里面的截图吧，看完是不是觉得我那一句话简单易懂点。

Algorithm 1 DEEPWALK(G, w, d, γ, t)

Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t

Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
Build a binary Tree T from V
for $i = 0$ to γ **do**
  $\mathcal{O} = \text{Shuffle}(V)$
  for each $v_i \in \mathcal{O}$ **do**
    $\mathcal{W}_{v_i} = RandomWalk(G, v_i, t)$
    SkipGram$(\Phi, \mathcal{W}_{v_i}, w)$
  end for
end for

3.item2vec与word2vec

i2v和w2v算法原理上是一样的，源于w2v，w2v基于句子序列抽取词与词之间的共现关系得到词的embedding，i2v是在w2v在推荐领域的应用，基于item序列来得到item的embedding，item可以是推的任何物品，如商品，视频，文章等等。

至于如何训练得到embedding，以及什么skip-gram和cbow，懒得写了

4.embedding

embedding就是向量，是实体在高维空间的数学表示，表示成embedding之后，就可以进行实体之的计算比较了。

二、deepwalk实战

其实实战时很简单的，就是调包了

talk is cheap，show me the code

1.基于用户点击数据构图

构图使用的networkx，集成了pandas的dataframe一键构图

```
    df = pd.read_csv('data.csv', index_col=0)
    df.columns = ['user_id', 'item_id', 'ts', 'dt']
    # 增加用户id和item_id的前缀已区分节点
    df['user_id'] = 'u' + df['user_id'].astype(str)
    df['item_id'] = 'i' + df['item_id'].astype(str)
    df.sort_values('ts', inplace=True)
    df.reset_index(drop=True, inplace=True)
     # 构造图模型，一句话的事（人生苦短，我用python）
    G = nx.from_pandas_edgelist(df, from_node, to_node, edge_attr=True, create_using=nx.Gr
ph())
```

2.随机游走产生序列

```
def get_randomwalk(G, node, path_length):
    """
    随机游走
    :param G:
    :param node: 当前节点
    :param path_length: 序列长度
    :return:
    """

    random_walk = [node]
    for i in range(path_length - 1):
    # 获取相邻节点
        temp = list(G.neighbors(node))
        temp = list(set(temp) - set(random_walk))
        if len(temp) == 0:
            break
        random_node = random.choice(temp)
        random_walk.append(random_node)
        node = random_node

    return random_walk
# 获取图的所有节点
all_nodes = list(G.nodes())
random_walks = []
# 遍历所有节点
for n in tqdm(all_nodes):
    for i in range(sample_count):
        random_walks.append(get_randomwalk(G, n, seq_len))
```

3.使用word2vec训练模型得到embedding

使用的是gensim.models里面的Word2Vec，简简单单三句话

```
    # 使用w2v训练embedding
    model = Word2Vec(size=10, window=5, negative=10, alpha=0.03, min_alpha=0.0007,
                     workers=40, seed=9482, sg=1, iter=5)
    model.build_vocab(random_walks, progress_per=2)
    model.train(random_walks, total_examples=model.corpus_count, epochs=5, report_delay=
)
```

然后就得到所有的item和user的embedding了，就可以用各种相似度算法计算物品之间的相似度了

三、工程化

1.预存topN直接使用

预存topN是预先计算每一个item的最相似topN个item，然后存在数据库里面供线上直接调用。

```
    # 获取item_id
    data=df.drop_duplicates('item_id')[["item_id"]].reset_index(drop=True)
    # 获取每一个item的embedding
    for i in range(0, 10):
        col = "itemid_{}_deepwalk".format(i)
        data[col] = data['item_id'].apply(lambda x: model.wv[x].tolist()[i])
    # model里面只保留item的embedding
    restrict_w2v(model.wv,data.item_id.tolist())
    # 获取每一个item的相似topN
    data['sim_list'] = data['item_id'].apply(lambda i: [x[0][1:] for x in model.wv.most_similar(f'{i}',
opn=50)])
    data['item_id'] = data['item_id'].apply(lambda x: int(x[1:]))
    data['sim_list'] = data['sim_list'].astype(str)
    data.to_csv('data/video_id_embedding_content_level_4_5.csv')
    # 存入数据库供业务端使用
    rows = data.to_sql('ads_ft_short_video_deepwalk_embedding', get_engine_223_tidb5_for_fe
ture(), index=False,if_exists='replace')
```

关于删除word2vec中embedding，在实际中还是蛮有用的，在混合了用户和item的embedding里需要洗出item的embedding，然后再借助w2v的api most_similar直接计算topn

```
def restrict_w2v(w2v, restricted_word_set):
    new_vectors = []
    new_vocab = {}
    new_index2entity = []
    new_vectors_norm = []

    for i in range(len(w2v.vocab)):
        word = w2v.index2entity[i]
        vec = w2v.vectors[i]
        vocab = w2v.vocab[word]
        vec_norm = w2v.vectors_norm[i]
        if word in restricted_word_set:
            vocab.index = len(new_index2entity)
            new_index2entity.append(word)
            new_vocab[word] = vocab
            new_vectors.append(vec)
            new_vectors_norm.append(vec_norm)

    w2v.vocab = new_vocab
    w2v.vectors = new_vectors
    w2v.index2entity = new_index2entity
    w2v.index2word = new_index2entity
    w2v.vectors_norm = new_vectors_norm
```

2.faiss构造embedding索引，实时查询相似topN

faiss是一个Facebook AI团队开源的库，全称为Facebook AI Similarity Search，该开源库针对高维间中的海量数据（稠密向量），提供了高效且可靠的相似性聚类和检索方法，可支持十亿级别向量的索，是目前最为成熟的近似近邻搜索库。

我们直接存embedding在faiss库里，供线上使用。线上实时计算相似topN

关于线上使用方法：

- 基于i2i召回物品的相似物品

可以直接获取用户点击的物品embedding，然后计算点击物品的topN个相似物品

- 基于u2i召回用户相似物品

获取用户的embedding，计算和用户embedding相似的topN个item