基于 skip-gram 的阅读内容推荐技术探索

唐佳琪[1]，郭星宇[1]，张欣蕾[1]

（华东师范大学，上海市 200062）

摘要：在 CCIR2018 评测中，我们队伍通过数据特征和场景分析，探索将 Skipgram 嵌入降维模型运用到了网上阅读社区的文本内容推荐场景中。在具体实现中，利用用户交互行为历史的文章 ID 等作为训练语料，并融合交互历史，作为相关输入，最终得到最相近的文章，从而对用户进行个性化推荐。与一些其他方法进行对比，以离线和在线测试分数作为评价指标， skipgram 模型显著优于其他实现，用于推荐能够达到较好效果。

关键词：推荐系统；fasttext；知乎

Abstract: In CCIR2018 contest, our team applied skipgram model in NLP to recommender system. We used answer ids from interaction of user-answer history as words to train corpus. Then we used interaction history from testing dataset as sentences to find the most similar answers to recommend. We compared this method to other methods and used offline and online results as assessment criteria. After some experiments, we found that skipgram model could yield good results for recommendation.

Keywords: recommender system; fasttext; Zhihu

1 介绍

本次任务的主题是在移动环境下知识分享平台“知乎”上的内容推荐。参赛者需要在给定用户、历史阅读序列及其相关属性的情况下，将合适的内容推荐给用户。测评分成离线测评和在线测评两部分。离线测评使用 NDCG 作为评价指标，在线测评使用点击率 CTR 作为评价指标。

在离线测试阶段，我们尝试过使用 XGBoost 计算各属性权重并对每个话题下的文章按热门程度（评论数、点赞数等）排序，根据用户阅读记录推荐相应话题下热门文章；对话题进一步聚类，减少话题数量；使用 TensorRec，输入文章向量、用户向量及用户文章交互矩阵，输出用户文章排名矩阵；使用 HIN 异构信息网络（如用户到话题，话题到文章为一条路径，计算用户到每篇文章的路径数目并排序）；使用 spotlight 库，输入用户、文章、阅读时间三个序列，输出用户对文章评分；使用 word2vec、fastText 对用户阅读记录的 ID 列表训练模型，输出与用户阅读相似的文章进行推荐。

在线测试阶段，我们主要使用了 fastText 模型，并尝试将多种方法的结果按比例融合。本文主要介绍了使用 fastText 模型的解决方案。

fastText 模型主要使用了 skip-gram[1]方法，该方法主要利用上下文信息找到单词的向量，使得意思相近的单词向量相似度高。

在将其用作内容推荐时，我们根据用户阅读历史计算出文章向量，使得相似文章向量相似度高，接着根据每个用户阅读行为以及文章向量表示找到与该阅读行为中相似的文章进行推荐。

在知乎文章推荐的场景下，我们发现文章特征庞杂，基于此我们对文章使用 xgboost 进行特征抽取。由于离线测试和在线测试场景的不同，我们针对不同场景对重复文章做了不同处理，并且针对在线测试时间紧张问题，我们在训练模型时进行了多线程处理，图 1 是离线场景下将 skip-gram 方法用作内容推荐时的流程图，图 2 是在线场景下将 skip-gram 方法用作内容推荐时的流程图。

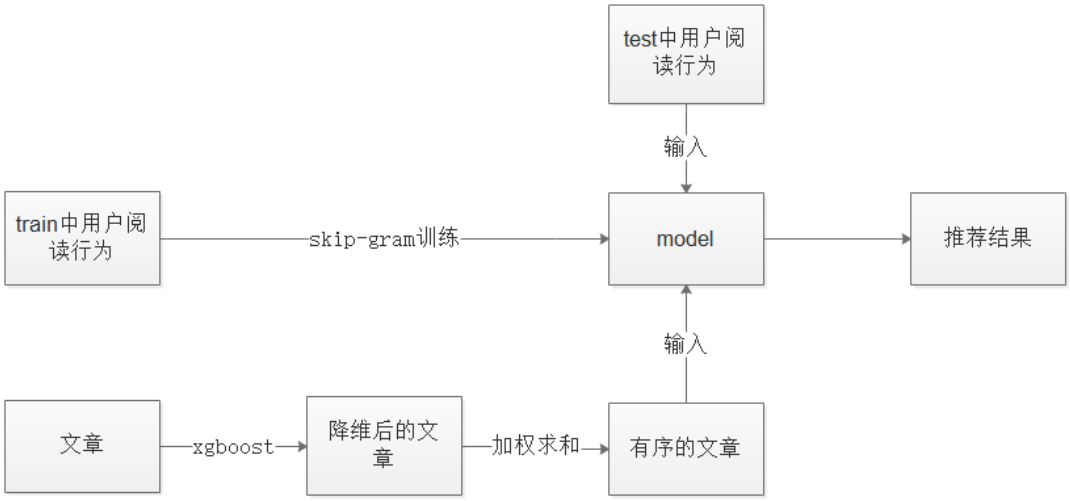

图 1：离线场景下基于 skip-gram 的内容推荐流程图

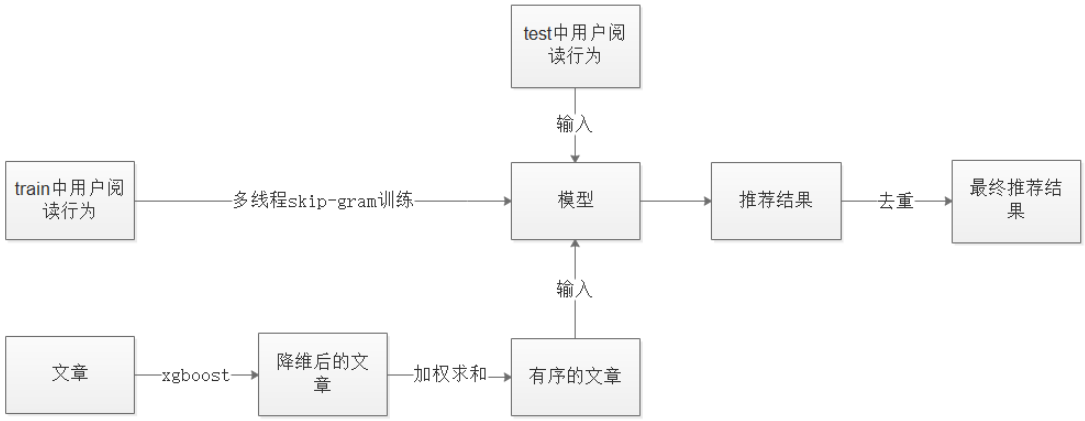

图 2：在线场景下基于 skip-gram 的内容 tuijian 流程图

2 不同特征的处理

2．1 训练集和测试集—用户文章交互历史

我们所使用的所有方法都用到了该数据。在此数据中，我们可以得到用户与一些文章有无交互、文章时序信息以及用户对不同文章的偏好。对于这些特征，我们尝试了将其训练为 word2vec/fasttext 中的语料，或者利用协同过滤的思想找出文章相似度、用户相似度，再进一步进行推荐。

2.2 文本信息

我们曾经将所有文章的文本信息训练出了语料，并试图利用 word2vec 找出和搜索词有关联的文章。但此种方法最终不可行，因为搜索词通常未在文本中出现、或者不能正确分词，导致找到的文章并没有关联。

2.3 文章属性

我们使用分类模型 xgboost，将训练集中用户与文章交互的阅读时间是否为 0 作为文章类别，answer_infos.txt 中的部分离散特征（是否被标记为优质答案、是否被编辑推荐、答案被赞同的次数等）作为属性，训练模型。得到各属性的权重，根据权重以及每篇 candidate 的各属性值，为每篇 candidate 计算分数，并从高到低排列。计算权重时，原先数据中的噪

声被有效地减小了。文章分数越高，该文章越热门。此分数在冷启动用户的推荐中比较重要。

2.4 用户属性

在 tensorrec 方法中，我们使用到了用户离散属性，和文章离散属性一起直接数值化，作为 tensorrec 的输入。实验证明，此种处理方法过于简单也不太适用。

在整个评测中，我们用到的特征还是比较少的，主要就是训练集、测试集和文章属性。如果我们成功的运用更多特征，并且将原数据噪声减小，相信结果会更准确一些。

3 模型实现

3．1 word2vec skipgram 模型

在推荐中主要使用了 skipgram 模型，该模型是 Mikolov 等在 2013 年[1]提出来的，该模型以一个单词作为输入，以该词的多个上下文词语做为输出。[2],[3]提出了更为高效的计算 softmax 的方式来改进 skipgram 模型，即 Hierarchical Softmax。该模型比较成熟的应用为 word2vec 方法。

评测中，我们使用用户阅读历史作为语料训练 Word2vec 模型。推荐时对于每个用户的每一篇历史阅读文章找到最相似的一篇 candidate 作为推荐文章。

此种方法利用训练集不断增加的数据规模、简单的模型，可以在离线测试中取得不错的结果。

但此种方法并不准确。首先我们仅仅使用了用户阅读历史 id 作为输入，没有用到文章或是用户的其他特征数据。其次对于出现在测试集但未出现在训练集中的历史记录，此方法无法找到相近的文章。同时，我们对于选取的文章顺序未做很多改进。开始时利用训练集的顺序取满一百篇为止，之后将阅读记录随机排序后再取满 100 篇。在线测试每个人只推荐 10 篇，顺序就显得很重要。不过因为此种方法包含在 fasttext 方法中，就并未单独得到结果。最后，对于冷启动用户也即没有任何阅读历史的用户，此种方法并不适用。

3．2 fasttext skipgram 模型

相比 word2vec，[4]中提出 subword 信息可以改进语言的向量表示，通过使用以字母为单位 n-gram 来表示语料库中没有出现过的词语，使它们可以表示字母级别的信息。此种模型也即我们之后采取的 fasttext。

评测中，我们同样使用用户阅读历史作为语料训练 fasttext 模型。不过不同的是，推荐时取出用户与回答的交互（去除用户与问题的交互，以及文章阅读时间为 0 的交互），进行 shuffle 操作后，整个作为 positive 参数调用 fastText 模型的 most_similar 方法。

此种方法相比第一种 word2vec 方法有很大的改进。首先，推荐文章的顺序是由整个阅读历史决定的，而不是像之前随机选择一篇文章。其次，未在训练集中出现的阅读历史也可以找到相近的文章。

但是同 word2vec 方法一样，此种方法也有特征选择太少、冷启动的问题。同时，对于新的文章，利用 subword 信息并不像语言中不同形态变化的词语一样能准确得到相近文章。在线测试时，如果当天测试集中有较多新文章，很难找满最相近的 10 篇文章推荐给用户。此种方法也很耗时。

3．3 根据用户阅读过的文章话题,推荐该话题下的文章

此种方法是我们最早尝试的方法。利用话题将用户和文章建立联系并进行推荐。

我们认为用户阅读过的文章所关联的话题中，关联过越多文章，就代表用户对此话题最感兴趣。同时找出每个话题下热门文章推荐给用户。

此种方法可以一定程度表现出用户的阅读偏好，并推荐相关文章。

但此种方式弊端也很多。首先是有的用户阅读过的话题都比较相似，那么推荐的文章类型就比较单一，而且和阅读历史的风格很相似；有的用户阅读过的话题很杂，那么排序也是

个问题。其次此种方法下，冷门文章并不会被推荐出去，推荐的都是人气高的文章。最后是话题的细粒度并不大，可能有的大话题嵌套小话题，但也会作为不同的话题进行推荐。

3．4 HIN 异构信息网络[5]

认为用户到文章存在一条路径，根据用户、主题、文章的关联，计算出用户到每篇文章的路径数目并进行推荐。

此种方法可以表现出用户与每篇文章的关联程度，我们认为关联越多，用户越有可能阅读此篇文章。

但是此种方法也有一些弊端。首先是用到的特征比较少。其次是对于冷启动问题也不能解决。

3．5 Spotlight 库

该方法主要利用了时序信息,使用神经网络训练模型,接着以用户阅读文章为模型输入,得到用户对文章评分。

此种方法不同于之前方法的是利用到了时序信息。

但此种方法同样有特征选取过少以及冷启动的问题。而且因为是直接调用 spotlight 库，我们并没有修改参数调到最佳。同时其中用到了 CNN 来训练模型，对时间和资源消耗都比较大。

3．6 Tensorrec 库

将文章属性、用户属性以及用户对部分文章的评分输入，得到用户对所有文章的输出。此种方法用到了较多离散特征。

但此种方法的得分是六种方法中最差的。可能有以下原因。一、我们对于输入的评分只是简单以-1,0,1 来表示。二、输入的特征仅仅简单地数值化，变成了连续的整数，噪声比较多，并且没有使用 one hot encoding。三、此种方法使用两个特征矩阵相乘后得到的矩阵再与用户文章交互的矩阵进行对比并在 tensorflow 上改进，其实很不准确。我们的数据中没有用户对文章的评分，用户与文章的交互只有 1,0,-1，两个矩阵相差很大，导致误差很大。总的来说，这种模型并不适合我们的应用场景。

4 实现技巧:

在比赛过程中我们主要使用了六种方法,我们将其归结为三种思路，并且将比赛中使用的方法以及对每种方法进行的优化总结如下,在本章节我们将重点介绍效果最好的 fasttext 方法实现流程及其独特设计.

4.1 方法流程

4.1.1 思路 1

从用户阅读历史找到这些阅读历史对应话题,和用户关注的话题一起认为是用户喜欢的话题,给该用户推荐该话题下的文章.基于该思路我们设计了一种实现方法,并且进行了三种不同的优化图 3 为该思路流程图。

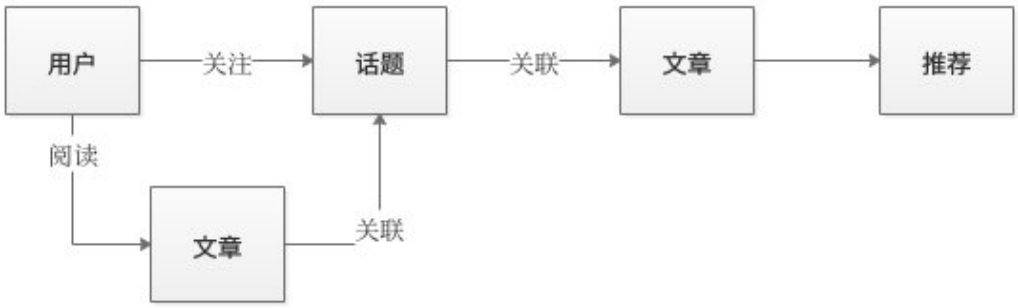

图 3 思路 1 流程图

方法 1：根据用户阅读过的文章话题,推荐该话题下的文章

对于所有（test）用户，找到他们阅读历史中所有 answer/question 对应的 topic，统计用户阅读每个 topic 的频数，将这些 topic 作为每个用户感兴趣的话题集，频数作为兴趣度。考虑到数据量大、运行时间长的问题，这里我们仅尝试过取出用户最后阅读的 5 篇，10 篇，15 篇，20 篇文章，效果最好的为 10 篇。

对 candidate 中所有 answer，找到其对应的所有 topic。然后将 answer 归类到相应的每个 topic 中。

对于 candidate 中每篇 answer，给其不同属性赋予不同的权重值，然后对每篇 answer 计算一个最终值，并进行排序。

（计算各属性权重的方法：使用分类模型，抽出 train 用户历史行为中文章阅读与否这个数据，然后将文章属性作为输入，文章阅读与与否作为输出训练一个分类模型，这里使用的是 xgboost）

对 test 中用户进行推荐的时候，根据之前已经找到的每个用户的感兴趣话题，用频数/total 算比例，按比例和分数从高到低顺序取出要推荐的各话题下的文章。如果未取满，则推荐 candidate 中得分最高的 100 篇文章。

改动 1.1：未取满，则先取用户关注话题下的高分文章，再取所有 candidate 中的高分文章，一定程度上改进了冷启动问题。

改动 1.2：调整 XGBoost 参数，使得准确率提高。

改动 1.3：对话题进行聚类，使用话题下的文章重合度计算相似程度，使用 kmeans 及层次聚类，使用聚类结果代替原先的话题。得到的结果较差，首先聚类效果差，即使进行了降维，仍然很不理想，其次给用户推荐的文章近似热门文章，聚类使得个性化和多样性都受到了影响。

4.1.2 思路 2

根据用户文章话题等的交互关系找到每个用户与文章的相似度,按照相似度从大到小进行推荐.基于该思路，我们设计了三种实现方法，图 4 为思路 2 的流程图。

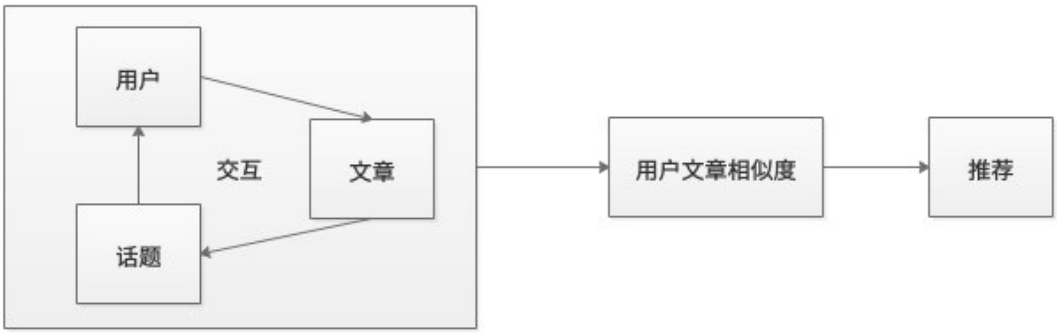

图 4. 思路 2 流程图

方法 2：TensorRec

使用 TensorRec，输入文章所有数值化属性、用户数值化属性、用户文章交互矩阵（0，1，-1），1 代表点击阅读,0 代表没有交互，-1 代表未点击即阅读时间为 0；直接调库得用户文章排名矩阵，按排名先后推荐文章。

方法 3：HIN 异构信息网络[5]

认为从用户到文章存在着一条路径，比如用户，文章，主题，文章这条路径。比如用户 U1 阅读了文章 A1，文章 A1 对应主题 T1，主题 T1 又关联文章 A2，那么用户 U1 到文章 A2 存在着一条路径，如果说用户阅读了 A3，A3 同样对应主题 T1，那么用户 U1 到文章 A2 的路径数目则加一。

使用这种方法设计了两条路径 UATA.用户阅读文章，文章关联话题，话题关联文章。UAUTA.用户阅读文章，用户阅读文章转置，用户关注话题，话题对应文章。

再按照用户到每篇文章路径数量由高到低排序，推荐文章。

方法 4：spotlight

该方法主要利用了时序信息,使用神经网络训练模型,接着以用户阅读文章为模型输入,得到用户对文章评分。

首先找到用户阅读的文章，及阅读文章的时间，生成用户，文章，时间，三个 list。

以这三个 list 作为输入，使用 cnn 方法训练模型。

得到模型以后可以将用户阅读行为进行输入,最后该模型会根据输入,对每篇文章进行评分。根据分数高低推荐文章。

4.1.3 思路 3：

根据用户阅读历史计算出文章向量,使得相似文章向量相似度高,接着根据每个用户阅读行为以及文章向量表示找到与该阅读行为中相似的文章进行推荐.基于该思路，我们设计了两种实现方法，图 1 为思路 3 方法流程图。

方法 5：word2vec

将训练集中用户的阅读历史和样例文章中的 answer id 抽取出来，以空格分隔，注意去除没有点击阅读的（即阅读时间为 0）以及类型为 Q 的（即问题，此处只考虑回答）文章，作为一句句子。整个训练集整理完的句子合集作为语料，训练 word2vec 模型。

将测试集中用户的阅读记录抽取出来，同样删去阅读时间为 0 的以及类型为问题的文章。对于每一个用户，首先查看是否有阅读记录，如果有，对每一篇历史阅读文章找到最相似的一篇 candidate，放到结果集中。如果没有取满 100 篇，则考虑用户关注的 topic 集合中的高分文章以及所有高分文章。

方法 6：fastText

对训练集进行处理，取出每条样本中的文章的 ID，以及阅读该条样本中的文章之前，用户与其他文章（回答）的交互中的 DocID（去除用户与问题的交互，以及文章阅读时间为 0 的交互）。对取出的 DocID 的列表进行 shuffle 操作，并写成一行用空格隔开的 ID 序列（去掉仅一条文章的行），整个 training_set.txt 处理成一个 txt 文件（train_read_history_part.txt）。

目前所使用到的训练集：离线测试时 training_set.txt (78G)、old_testing_set.txt (3.8G)、testing_set.txt（3.8G)、在线测试时每天 21 支队伍的 testing_set.txt(每个文件 43.5M,合并后文件 914.5M) （每天持续增加）

使用 train_read_history_part.txt 训练 fastText 模型。

取出 user_info.txt 中用户关注的话题 ID 列表。

使用分类模型 xgboost,将训练集中用户与文章交互的阅读时间是否为 0 作为文章类别，answer_infos.txt 中的部分信息（是否被标记为优质答案、是否被编辑推荐、答案被赞同的次数等）作为属性，训练模型。得到各属性的权重，根据权重以及每篇 candidate 的各属性值，为每篇 candidate 计算分数，并从高到低排列。

对于测试集，取出用户与回答的交互（去除用户与问题的交互，以及文章阅读时间为 0 的交互)，进行 shuffle 操作后，作为 positive 参数调用 fastText 模型的 most_similar 方法，尽量取满 10 篇相似的 candidate 作为推荐文章。如果不能取满，根据用户关注的话题，取话题中的高分文章作为推荐。最后如果还没有取满，直接使用高分文章作为推荐。

图 5 为该方法的代码流程图。

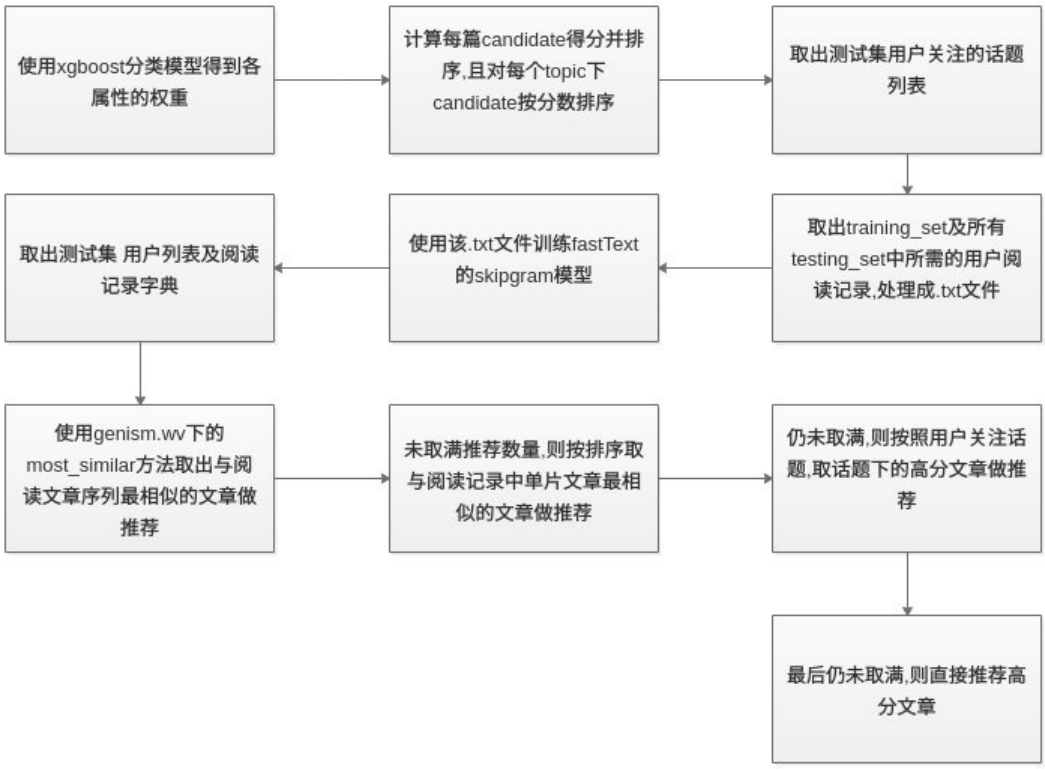

图 5. fasttext 方法代码流程图

4.2 独特设计:

skip-gram 模型是 word2vec 的一种,主要用于获取词向量，使得同一上下文的词对应词向量相似度高。在将 skip-gram 用到文章推荐的时候,我们将每篇文章 id 看做一个单词，用户的阅读行为看做单词的上下文,认为一个用户读过的文章间相似度高。基于这种思想,使用

神经网络构造模型计算出的文章向量越相似，则可以认为该文章与用户的阅读行为越相关，进而将与用户阅读过的文章最相似的文章作为推荐。

5 结果对比

5．1 离线测试

在前面的章节我们已经详细地介绍了各种方法及实现技巧。其中效果最好的就是 fasttext 方法和 word2vec 方法，都是利用用户阅读历史，计算文章向量表示，并将此训练成语料进行推荐。Spotlight 方法利用用户阅读文章的时序信息，构建了 deep and shallow 推荐系统。Hin 异构网络方法、tensorrec 方法利用用户与文章交互历史、用户和文章的特征，找出用户文章的相似度进行推荐。话题方法则是利用用户文章交互历史，找出用户的兴趣倾向进行推荐。

5.1.1 结果对比

方法 1：fastText

方法 2：word2vec

方法 3：spotlight

方法 4：TensorRec

方法 5：HIN 异构信息网络（UATA）

方法 6：根据用户阅读过的文章（最后 10 篇）话题,推荐该话题下的文章

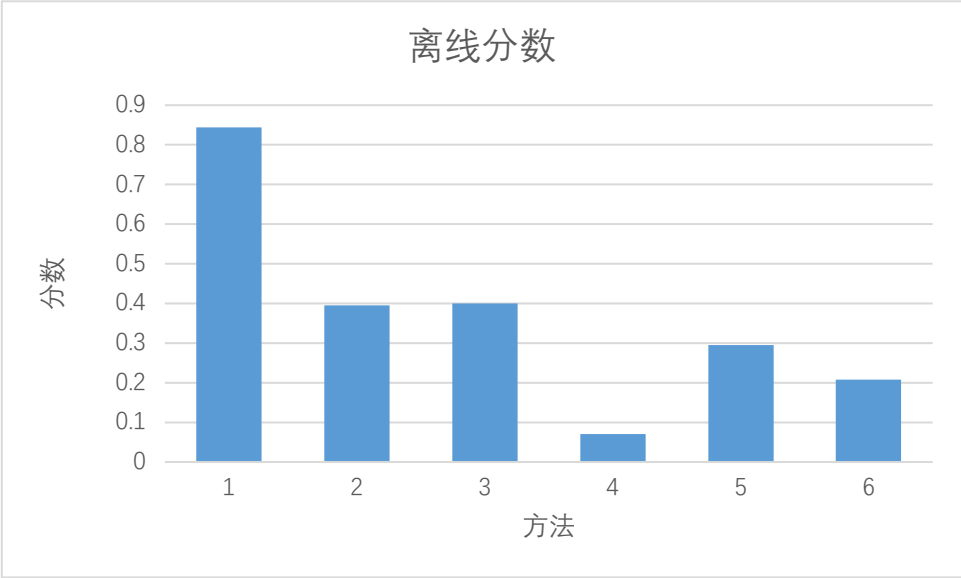

图 6. 离线测试阶段各方法结果对比

5．2 在线测试

由于数据每日更新且提交结果时间有限，用于训练 fastText 模型的 txt 文件随着每日 test 文件中的阅读记录增加而逐渐增大，模型训练时间很长，面对来不及训练新的模型的情况，我们尝试直接使用前一天的模型，并删去少于两篇文章的阅读记录；提取高频用户的阅读记录；增大 fastText 模型训练的线程数，减少了模型训练时间。

在此期间，我们也尝试了使用多种方法的结果按比例融合，得到最后的推荐结果。

5.2.1. 结果对比

每日为每位用户推荐十篇文章：

2018.7.30-2018.8.3、2018.8.5、2018.8.8-2018.8.10：fastText
2018.8.4：HIN（UATA）两篇、fastText 八九篇
2018.8.6：fastText 四篇、spotlight 两篇、HIN（UATA）四篇
2018.8.7：方法 1（话题）三篇、HIN（UATA）三篇、 fastText 四篇

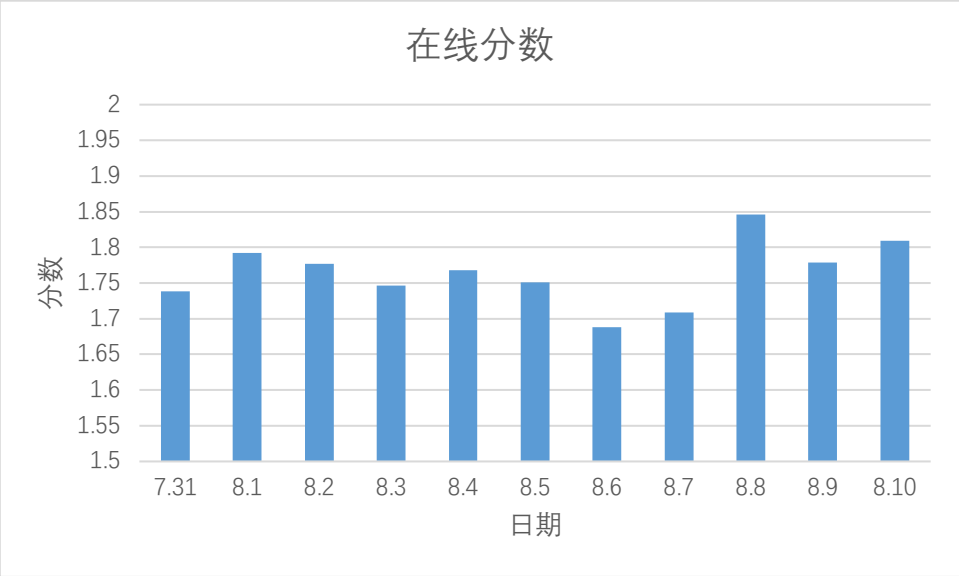

图 7. 在线测试阶段每日分数对比

6 总结

比较各种方法的结果好坏，我们发现根据用户的偏好推荐热门文章是不够的，用户需要更个性化的服务，同时多样性是不可缺少的，我们应该尽可能地平衡这两点。在此期间，我们注意到 ID 是很重要的一个属性，使用 ID 进行 embedding 得到的结果能较好地满足个性化需求，一定程度上避免了只推荐热门文章的问题，在实际应用中的效果也很不错。

另外，冷启动问题是此次比赛的重点之一，遗憾的是，我们目前只是使用新用户关注的话题为他们推荐该话题下的热门文章，还没有找到更好的解决方案。我们也期待对模型进一步理解和改进。

在比赛中我们将用户阅读行为作为 skip-gram 模型的上下文取得了较好的推荐效果，这种思想同样适用于其它场景，例如社交网络中的推荐，基于用户关注列表进行好友推荐。不仅如此，在电商平台，根据用户购买列表能够对用户进行商品推荐。除推荐以外，该模型能够进行同义词挖掘，用来分析用户关系等。因为 skip-gram 模型能够得到物品的向量表示，该模型能够对物品进行建模并作为其他模型的输入，利用向量表示同样能够达到快速检索的目的。向量表示以及上下文信息的融合使得 skip-gram 模型能够适应诸多场景。

参考文献：

[1] Mikolov, Tomas, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013a.
[2] Frederic Morin and Yoshua Bengio.Hierarchical probabilistic neural network language model. In Proceedings of the international workshop on artificial intelligence and statistics, pages 246–252,2005.
[3] Mnih, A., & Hinton, G. E. A scalable hierarchical distributed language model. NIPS (pp. 1081–1088),2009.
[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606, 2016.
[5] Zhao,H.Yao,Q.Li,J.Song,Y.and Lee,D.L.Meta-graph based recommendation fusion over heterogeneous information networks.In KDD,635–644,2017.

作者联系方式：
唐佳琪　华东师范大学中北校区 200062　15317785732 Jasminetang1231@126.com
郭星宇　华东师范大学中北校区 200062　15317563173 10152510116@stu.ecnu.edu.cn
张欣蕾　华东师范大学中北校区 200062　15201702756 10152510178@stu.ecnu.edu.cn