架构师

ARCHITECT

特刊

推荐系统[实践篇]

SPECIAL ISSUE

May,2016

架构师特刊

InfoQ

目录

序言

推荐系统从诞生的第一天起，就是在产业界需求的伴随下不断孕育和成长。不论是最朴素的思路，还是高深晦涩的算法，实践运用中的效果，才是推荐系统最核心的关注点。因此无论是何种理论或架构，一定是要经受住前线的炮火锤炼，才能经久不衰。

通过这期技术特刊，我们总结了工业界在推荐系统的实践运用中的各类经验，集中展示了如何将理论与实践结合，开发出一个效果优秀、运行可靠的推荐系统。

实践中，应用场景的多样性是最常遇见的问题——待推荐的物料在电商、媒体、教育、招聘等行业中各不相同，它既可以是商品，也可以是新闻、小说、歌曲、视频，等等……针对不同的运用场景，需要我们因地制宜的对算法和模型进行优化，这里有很多的经验与大家分享。

随时间动态变化的数据，也是需要处理的关键一环。用户的行为数据源源不断的产生并动态变化，设计良好的推荐系统，不能仅仅局限于静态的算法和模型生成，还需要有能力将动态的数据融合进来，快速的捕捉行为数据、动态的调整推荐结果、准确的把握推荐的时机，这里考验的是推荐系统的架构能力。

推荐系统架构的稳定性、延展性也是不容忽视的。与实验室的模拟环

境不同，生产环境的系统一旦上线运转后，不仅要承受每秒千万次的服务调用，还需要不断的进行扩展和升级，因此设计一个灵活可扩展的架构，才能不断吸纳新的算法和模型，同时保证系统长期稳定的服务。

推荐系统是通过机器学习技术，挖掘数据背后的规律，不断刻画用户需求，洞察人性的过程。真实世界里的数据是纷繁杂乱的、用户的兴趣是多样且变幻的，这也是充满挑战和乐趣的课题，只有在实践中反复锤炼，才能让推荐系统精益求精，充分的发挥价值、闪耀光芒。

知识是珍宝，而实践是开启它的那把钥匙。

陈运文 博士

达观数据 CEO

第 1 章 微博推荐架构的演进

引言

微博（Weibo）是一种通过关注机制分享简短实时信息的广播式社交网络平台。微博用户通过关注来订阅内容，在这种场景下，推荐系统可以很好地和订阅分发体系进行融合，相互促进。微博两个核心基础点：一是用户关系构建，二是内容传播，微博推荐一直致力于优化这两点，促进微博发展。如图 1 所示。

在微博推荐发展的过程中遇到体系方向的变化、业务的不断更迭、目标的重新树立，其产品思路、架构以及算法也随之进行变迁。本文主要阐述在这个过程中推荐架构的演进，从产品目标、算法需求以及技术发展等维度为读者呈现一个完整的发展脉络，同时也希望通过这个机会跟大家一起探讨业务与技术的相互关系。

为了便于理解微博推荐架构演进，在介绍之前需要陈述一下微博推荐在流程上的构成，其实这个和微博本身没有关系，理论上业内推荐所存在的流程基本都是相同的。如图 2 所示，推荐是为了解决用户与 item 之间的关系，将用户感兴趣的 item 推荐给他 / 她。那么，一个 item 被推荐出

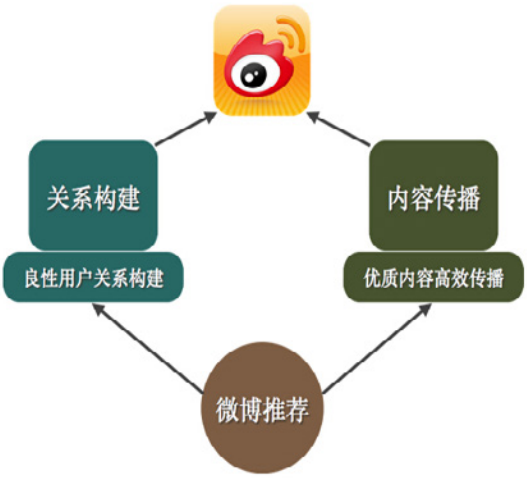

图 1 微博推荐的使命

图 2 推荐的链路

来会经过候选、排序、策略、展示、反馈到评估再改变候选等等形成一个完整的回路。

在上述整体流程的基础上，微博推荐架构经历了如图 3 所示的三个阶段。

通常架构的产生都会来自于团队和业务环境，源于环境因素而致力于解决环境中的问题，架构形成会带着较为强烈的特点，在其实施中会产生交给针对性的效果。本文将从环境因素、架构组成与特点以及实施效果这三个方面进行阐述微博推荐的三个阶段。

1. 独立式的1.0

1.1 环境

影响架构形成的环境因素可以分为内部环境因素以及外部环境因素。内部因素主要是团队及其成员相关内容，而外部因素主要来自于外部门、整个公司或者整个行业领域。

微博推荐 1.0 的这段时间是从 2011 年 7 月份到 2013 年 2 月份左右，其主要的目标就是实现当前的业务需求。对于独立式的解释：每一个业务项目都是一套完整架构流程，架构之间相对独立，甚至包括技术栈。之所

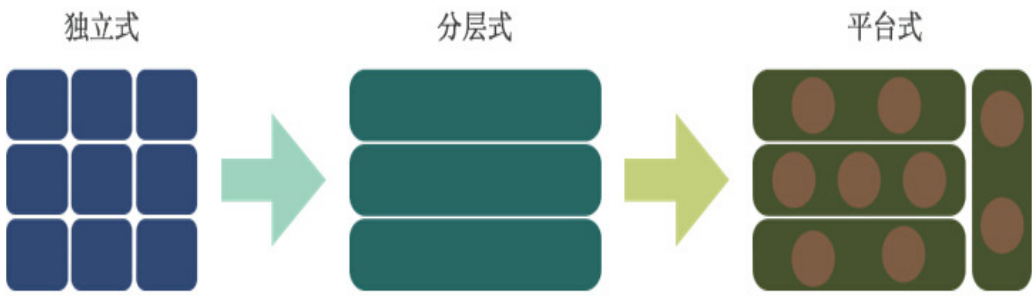

图3

以称之为独立式其内部因素有几点：

- 当时团队是一个新团队，成员也相对较新，相互的合作不多，缺乏推荐领域整体性经验。
- 团队成员对于推荐架构都有自己的一些或多或少的理解，但是对于在当前场景下的微博推荐架构，共识并没有形成。

当然起决定性因素的还是外部环境，是因为内部原因还是比较好协调和进化的。当时的外部环境因素如下所述。

- 项目需求很多，在当时一个5人团队并行开发的项目平均在3-5个左右，当然最重要的因素是当时的微博产品正处于高速发展期，很多地方都需要微博推荐的支撑。同时，项目周期也很短，排期仓促，很难有时间进行细致的整理和抽象。典型产品包括：微吧、微群、微刊、微话题、用户以及内容排序等。
- 团队是一个支撑性的，绝大部分需求来自于外部团队，各个外部团队不同的产品方向也导致疲于应付需求。
- 当时业内的推荐架构也有不同的发展方向，大家都在尝试摸索一些符合自身发展的架构思路。

由于上述的那些原因，通常我们面对一个接一个的项目时，都会根据自己的理解使用熟悉的技术栈来搭建流程，这样形成了一个又一个的独立架构。

1.2 架构组成与特点

上节中提到了独立架构形成的原因，大家可能觉得架构组成没有必要去描述了，这是不对的，事实上后来的分层以及平台架构的基础恰恰都来源于这个阶段，没有这个阶段团队不断踩坑总结就没有因地制宜产生的后续进化。因此，我们需要为大家剖析一下推荐 1.0 的架构组成与特点。

1）技术目标

参考图 2 所示，以业务实现为主要目标的微博推荐 1.0，没有建立起完整的反馈以及评估体系，同时排序也是被策略取代，那么讲主要的重点体现到了候选、策略以及展现上。上述推荐流程被转化为：候选 à 策略 à 展现简单形态。

2）架构组成

如图 4 所示，我们试图将每个项目的架构能够在图中表达出来，在真正的实施过程中，每一个项目负责人会选择使用 apache ＋ mod_python 作为服务架构同时，使用 redis 作为存储选型。在一些特定的项目中，引入了复杂运算从而诞生了 C/C++ 的服务框架 woo；同时，对于数据的存储要

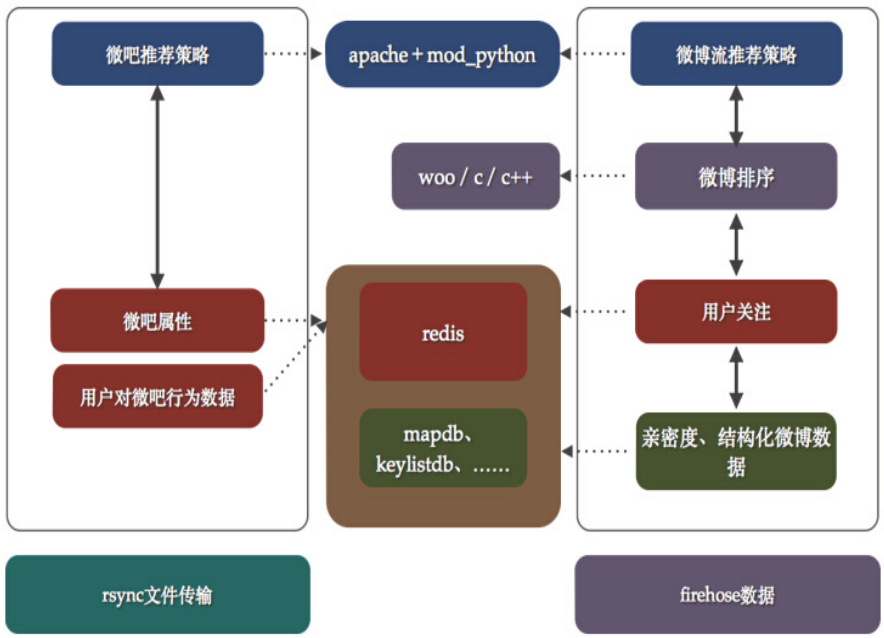

图4 微博推荐1.0架构简图

求特型化的项目中又自己研发了一系列的 db，比如早期存储静态数据的 mapdb，存储 key-list 的 keylistdb 等等。当然，在部署中会比下图更加随意一些，一个项目几台服务器部署好微博服务提供 http 请求，然后再找几个服务器安装 redis 作为数据支撑，来源数据和业务方定好规则使用 rsync 传输就 OK 了，大部分策略在 Python 中实现。

- Web服务：apache＋mod_python，后来发展成为社区更为完善的mod_wsgi。使用python作为WEB开发语言主要是因为平时处理数据使用的都是Python，同时上手快，学习曲线平缓。
- 运算服务：C/C++，形成woo内部服务框架。
- db：redis / mapdb / keylistdb等，分为两种存储方法：redis以及自研型。
- 数据来源：rsync文件传输，firehose作为微博相关内容来源）微博内部使用的一种数据队列）。

3) 架构特点

将架构特点划分为优点和缺点进行描述。那么优点是：

- 简单，易于实现，不需要额外的基础支撑
- 利于业务的功能快速实现
- 利于多业务并行开展，相互不影响

而不足是：

- 推荐流程不完整，缺乏反馈、评估等等重要内容，对于数据方面也极度缺乏统一处理方法
- 没有提供给算法相关的支撑，很难将推荐做的深入
- 几乎无法进行专业运维
- QA的测试仅仅能到功能层面，模块级别的测试几乎不可能，因为太过于分散
- 很难进行团队协作，不利于项目的分解

1.3 成果

尽管存在诸多的缺点，但是在其发展的过程中，也给后面的架构优化奠定了基础，其成果如下。

- 在微博高速发展的过程中，满足了微博对于推荐的业务支撑要求，在这段时期里面共完成二十多个独立项目。
- 诞生了woo的基础框架，后面的内部高效运算框架来源于此。
- 诞生了mapdb的静态存储，成为后期微博推荐静态存储的雏形。
- web应用层的不断需求的总结，组建形成推荐通用应用框架。

2. 分层式的2.0

上一节介绍完独立的 1.0，按照架构发展的道路，我们到了分叉路口，一边是流行的 LAMP 架构，另一边是符合广告、搜索的 CELL 架构。LAMP 架构数据策略分离，脚本语言作为业务开发主要语言，项目快速开发和迭代的首选。CELL 结构强调本地流程处理，数据与业务耦合性强，自研的服务以及数据库较多出现，适用于高性能效果型产品。最终我们选择兼容两者，倾向于业务的架构体系。为何如此呢？让我们再来看看当时的环境。

2.1 环境

微博推荐 2.0 的时间段是 2013 年 3 月份到 2014 年年底，这段时间内部环境因素如下：

1） 当前团队成员合作已经很长时间，彼此相互熟悉，同时对于技术选型有了一定的共识。

2） 团队产品进行了聚焦，针对内容 / 用户 / 垂直类三类推荐进行了整理，同时对于场景分别进行了重点划分：feed 流内、正文页以及 PC 首页右侧。这种聚焦有利于进行架构统一，同时也为技术争取了时间。

而外部因素是：

1） 公司对于推荐有了比较明确的定位，提高关系达成以及内容传播

效率，同时为推荐型广告打好技术探索、场景介入以及用户体验的基础。

2) 推荐领域里，各个公司都纷纷有了对于架构的产出，对于微博推荐有了很好的指导意义。

2.2 架构组成与特点

团队在执行核心业务实现的时候，不断演进工具以及框架，构建 2.0 的目标呼之欲出。

1) 技术目标

与 1.0 不同，仅仅实现业务需求已经不是 2.0 的技术目标了，针对完整的推荐流程，我们需要解决：

- 首先要实现完整的推荐流程，架构覆盖候选、排序、策略、展示、反馈和评估；
- 以数据为先，提炼出数据架构。实现数据对比，效果以数据为准；实现数据通道，体现反馈；实现数据落地，承接业务需求；
- 提供算法方便介入的方式；
- 既能保证业务的快速迭代和开发，又能支持高效运算。

2) 架构组成

微博推荐 2.0 的架构如图 5 所示，它不再是一个个独立的系统，也不是会让开发人员使用不同的技术解决相似的问题。这个架构图主要包括几个部分的内容。

- 应用层：主要承担推荐策略以及展现方面的工作，其特点在于充分发挥脚本语言的特点响应迭代需求。大部分的推荐内容经过排序之后已经可以展示了，但是由于前端产品策略的设定需要融合、删选以及重排操作，需要这一层来完成，在技术层面属于IO密集型的。在技术选型上，早期在原有apache＋mod_python基础上进行了框架开发产生了common_recom_frame。该框架面向的是二次开发者，基于此框架可以很好的实现推荐业务流程。该框架

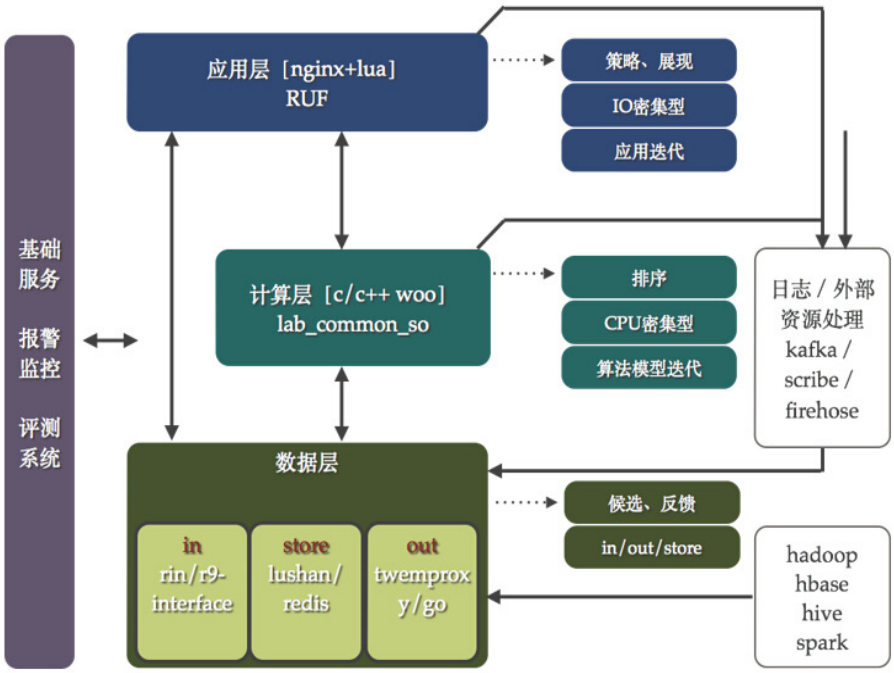

图5 微博推荐2.0架构示意图

的核心思想是提炼出project、work以及data的三层interface，project针对每一个推荐项目，work针对每个推荐项目中不同推荐方法，而data则是管理下游数据的访问方法。同时，设定了两个规范：一个是统一了推荐接口，无论是用户、内容还是垂直业务；另一个是屏蔽了不同协议数据库访问方法，极大提高了开发效率。common_recom_frame框架的诞生基本上解决了产品的各种推荐策略需求，走在了产品的前面。

- 计算层：主要承担推荐的排序计算，主要消耗CPU，在这一层给算法提供介入方法，支持算法的模型迭代。在这一层的技术选型上，我们继承了原有的WOO协议框架，一种基于C/C++开发的内部高效通讯框架。当然也做了不少扩展，依然借用了上面提到的common_recom_frame的思想，在WOO框架基础上实现了对于project / work/data的管理，提供给二次开发者更为高效的开发工具。在团队的开源项目中包含这个工具：https://github.com/

wbrecom/lab_common_so

- 数据层：主要承担推荐的数据流以及存储工作。数据层的工作主要是解决数据的IN/OUT/STORE问题。其中IN数据如何进入系统，OUT表示数据如何访问，STORE表示数据如何存。当在进行数据层规划的时候，又分析了微博推荐的数据特点，可以将其分为两类：静态和动态。静态数据的定义为： 更新需要全量同时频次较低的大规模数据；动态数据的定义为：动态更新同时频次较高的增量数据。这样在IN/OUT/SOTRE的大方向下，同时区分对待静态和动态数据，产生了RIN / R9-interface、redis/lushan、tmproxy / gout代理的工具或者框架。在这里展开讲一下，RIN支持数据动态数据的接入，通过web服务的方式接收数据，后端使用ckestrel进行队列管理，辅以多服务集群的消费框架，使用者只需要进行自己业务的开发即可快速上线消费动态数据。R9-interface处理静态数据的接入，推荐的大量静态数据来源于Hadoop集群的运算，r9-interface框架勇来解决静态运算［MR、HIVE SQL以及SPARK 运算］的通知、管理以及数据载入。针对推荐数据的存储，动态数据大量使用了redis集群，静态数据则使用了lushan集群。对于lushan这个工具在团队开源项目中也包含了：https://github.com/wbrecom/lushan。tmproxy / gout用来解决数据的OUT问题，gout是一个代理中间件，用来处理推荐中对于数据的动静结合访问的需求，减少业务对于后端数据变化带来的影响。
- 基础服务：推荐系统的基础服务主要包括监控、报警以及评测系统，数据监控系统分为性能以及效果监控两类，评测系统主要用来进行下线评估，在上线之前对效果有一定预期以及减少无效上线。图6展示了基础服务的UI。

图6 基础服务系统的UI

3）特点

优点是：

- 支撑完整的推荐流程，对于数据方面拥有统一的处理方法；
- 在兼顾业务功能快速实现的同时保证了效果技术的不断深化；
- 给算法提供了很好的支持；
- 提出以数据为先的思想，可以全面对比效果，推荐效果不断得以提升；
- 封层体系易于部署以及QA介入进行测试。

而不足是：

- 和推荐核心有一定的距离，并没有完全为推荐量身定做；
- 将推荐的策略算法完全交给了开发者，不利于推荐通用型；
- 对于算法的训练并没有涉及，仅仅是一个线上投放系统，不足以构成完整的推荐体系。

2.3 成果

微博推荐 2.0 的诞生产生很好的收益，其成果如下：

- 微博推荐的核心业务均在该体系下完成：正文页推荐、趋势用户推荐、趋势内容推荐、各个场景下的用户推荐、粉丝经济的粉条、账号推荐等产品。
- 诞生了lab_common_so的基础框架，并进行了开源。
- 诞生了静态存储集群解决方案lushan，并进行了开源。
- RUF框架的诞生极大提升了业务生产效率，同时也为openresty社区做出一定贡献。

3. 平台式的3.0

上节中描述 2.0 的时候提到了一个重要不足是“和推荐核心有一定的距离，并没有完全为推荐量身定做”，我们希望能够在推荐 3.0 中解决它，这个不足会带来什么问题，以及为何在已经满足业务需求的同时推荐的架构再次往前发展呢？那么接下来为各位展现微博推荐平台式的 3.0 设计，我们还是先看看所处的环境。

3.1 环境

微博推荐 3.0 的时间段是 2014 年底至今，当前的内部环境因素是：

- 推荐产品不在扩张，对效果更为看重，将工作重点从业务开发和迭代转化为以效果为目标的技术迭代。
- 新项目或者迭代推荐业务的时候发现重复的事情很多，而架构没有解决，工作存在冗余。

而外部因素是：

- 公司也从业务扩展转变为效率为先，提升用户体验以及内容质量上来。
- 微博推荐在推荐技术环节距离领域内有一定距离，当下有条件进行追赶。

3.2架构组成与特点

当前的环境也能体现出3.0的技术目标：

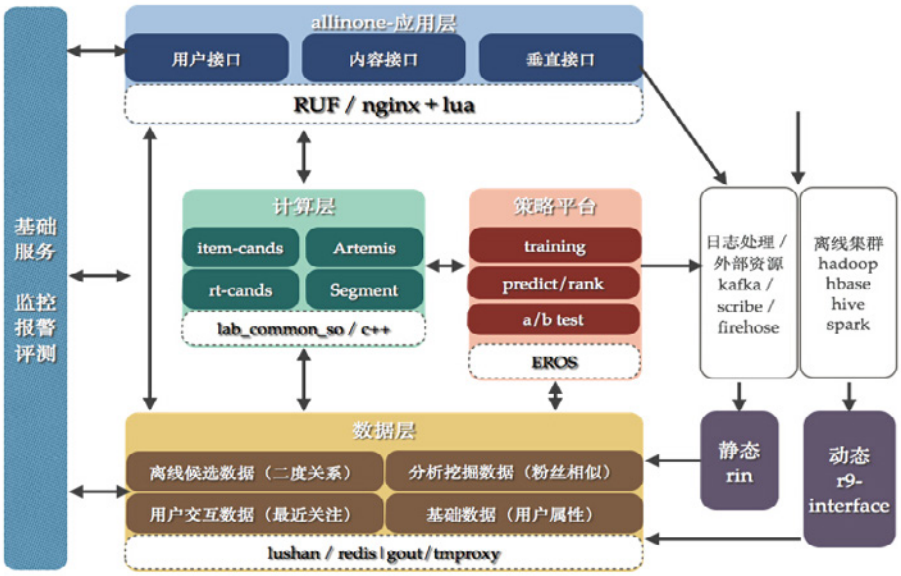

图7 微博推荐3.0的架构示意图

1）技术目标

与 2.0 不同，全覆盖推荐流程已经不是 3.0 的目标，其目标是：

抽象出推荐流程中对于候选 / 排序 / 训练 / 反馈的通用方法来

推荐是一个算法数据问题，应该以一个算法的角度构建推荐系统，因此需要更为贴近算法策略

2）架构组成

如图 7 所示，是微博推荐 3.0 的架构，也是当前实行的架构体系，大家其实可以发现，这是基于 2.0 发展起来的，既然还保留了大量 2.0 中使用的分层体系以及工具框架。在这里重点描述几个差异。

- 两个标准：一个是针对应用层，作为整体框架输出，应用层设定 all in one 接口标准，其标准包含了输入以及输出参数；另外一个是针对动态输入rin，由于离线计算我们可以确定结构，因此一个输入层工具r9-interface不需要设定规范，但是rin是需要进行标准设定，从属性 / 交互数据 / 日志等等层面进行划分。
- 计算层增加对于候选的标准生成方法：Artemis内容候选模块，item-cands用户候选模块、……，在项目开发中只需要选择这些

候选生成方法即可。

- 增加了策略平台EROS，解决算法模型的问题。EROS主要的几个功能是：1）训练模型 2）特征选取 3）上线对比测试。
- 数据层中的r9-interface以及rin增加对于候选的生成方法，在线以及离线使用推荐通用策略生成结果。

3）特点

主要描述其优势：

- 继承了原有2.0的特点，保留了其优势
- 对于推荐理解更为深入，结合更为紧密
- 解决了推荐候选 / 排序 / 训练的算法最重要问题

3.3 成果

微博推荐 3.0 的诞生，其成果如下：

- 微博推荐的核心业务会逐步迁移到该体系下，以算法数据作为驱动，提升效果。
- 诞生了EROS的训练流程，提出了训练的标准方法。
- 针对推荐设定了标准的输入输出方法。
- 针对候选，产生了具有抽象意义的推荐方法集合。

4. 总结

上文中对微博推荐架构演进做了较为详实的介绍，在这个演进的过程团队以及个人收益很大，技术与业务的关系在架构中得到了很好的体现。有以下几点可以跟大家分享。

1） 技术来源于业务同时提升业务发展，业务发展又反过来推动技术的前进，他们是一个相互影响相互促进的关系。和业务共同发展的技术才是有生命力的。

2） 技术架构的选型建议是寻找当前最短路径，然后进行不断优化迭代，一口气吃撑是不现实的，也是不合理的。

3） 推广某个框架和工具最好的方式不是行政命令也不是请客吃饭，而是的大家都是参与者，如同开源项目，每个人都是它的主人，这样人人维护，人人使用。

4） 团队崇尚简单可依靠，它说起来容易做起来难，不过有一个好方法就是懂得自己不应该做什么，而不是应该做什么。

5） 说到推荐这个特殊领域上来，设定目标，跟踪目标很重要，把数据和目标摆出来，产品、架构以及算法都会想办法去解决的。

作者简介

王传鹏，新浪微博商业产品部门推荐广告技术总监。2011 年加入微博，主要负责微博推荐广告技术以及算法平台的建设。曾于 2009 年创办过云信科技研发网络硬盘——99 盘，2010 年加入当当负责广告系统。

第 2 章 Netflix 公布个性化和推荐系统架构

Netflix 的推荐和个性化功能向来精准，前不久，他们公布了自己在这方面的系统架构。

2013 年 3 月 27 日，Netflix 的工程师 Xavier Amatrain 和 Justin Basilico 在官方博客发布文章，介绍了自己的个性化和推荐系统架构。文章开头，他们指出：

要开发出这样的一个软件架构，能够处理海量现有数据、响应用户交互，还要易于尝试新的推荐方法，这可不一点都不容易。

接下来，文章贴出了他们的系统框架图（图 1），其中的主要组件包括多种机器学习算法。

他们这样解释其中的组件和处理过程：

对于数据，最简单的方法是存下来，留作后续离线处理，这就是我们用来管理离线作业（Offline jobs）的部分架构。计算可以以离线、接近在线或是在线方式完成。在线计算（Online computation）能更快地响应最近的事件和用户交互，但必须实时完成。这会限制使用算法的复杂性和处理的数据量。离线计算（Offline computation）对于数据数量和算法复杂度限制更少，因为它以批量方式完成，没有很强的时间要求。

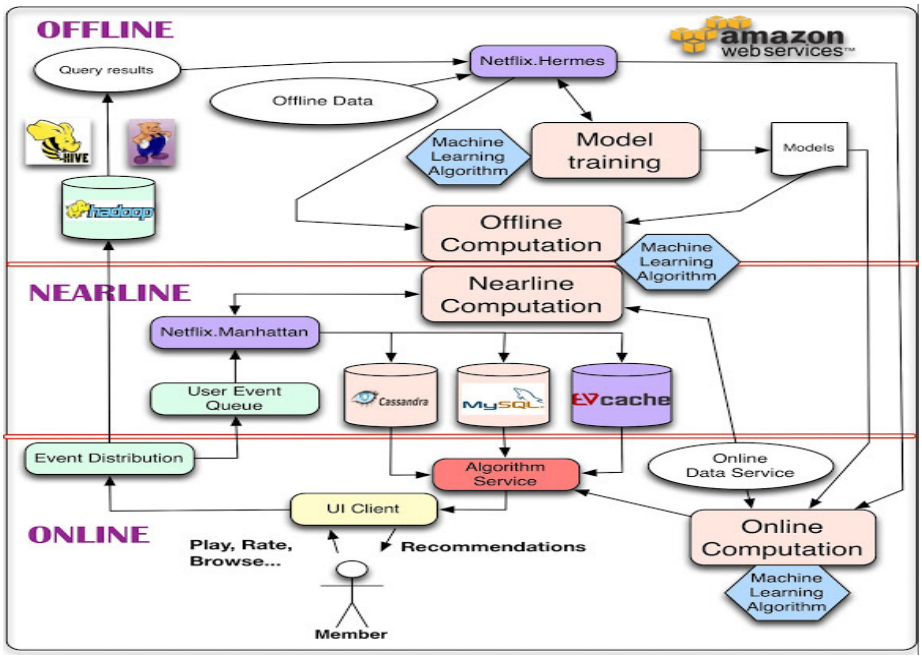

图1

不过，由于没有及时加入最新的数据，所以很容易过时。个性化架构的关键问题，就是如何以无缝方式结合、管理在线和离线计算过程。接近在线计算（Nearline computation）介于两种方法之间，可以执行类似于在线计算的方法，但又不必以实时方式完成。模型训练（Model training）是另一种计算，使用现有数据来产生模型，便于以后在对实际结果计算中使用。另一块架构是如何使用事件和数据分发系统（Event and Data Distribution）处理不同类型的数据和事件。与之相关的问题，是如何组合在离线、接近在线和在线之间跨越的不同的信号和模型（Signals and Models）。最后，需要找出如何组合推荐结果（Recommendation Results），让其对用户有意义。

接下来，文章分析了在线、接近在线和离线计算。

对于在线计算，相关组件需要满足 SLA 对可用性和响应时间的要求，而且纯粹的在线计算在某型情形下可能无法满足 SLA，因此，快速的备用方案就很重要，比如返回预先计算好的结果等。在线计算还需要不同的数据源确保在线可用，这需要额外的基础设施。

离线计算在算法上可相对灵活，工程方面的需求也简单。客户端的SLA响应时间要求也不高。在部署新算法到生产环境时，对于性能调优的需求也不高。Netflix利用这种灵活性来完成快速实验：如果某个新的实验算法执行较慢，他们会部署更多Amazon EC2实例来达成吞吐处理目标，而不是花费宝贵的工程师时间去优化性能，因为业务价值可能不是很高。

接近在线计算与在线计算执行方式相同，但计算结果不是马上提供，而是暂时存储起来，使其具备异步性。接近在线计算的完成是为了响应用户事件，这样系统在请求之间响应速度更快。这样一来，针对每个事件就有可能完成更复杂的处理。增量学习算法很适合应用在接近在线计算中。

不管什么情况，选择在线、接近在线、还是离线处理，这都不是非此即彼的决策。所有的方式都可以、而且应该结合使用。 …… 即使是建模部分也可以用在线和离线的混合方式完成。这可能不适合传统的监督分类法（supervised classification）应用，因为分类器必须从有标记的数据中批量培训，而且只能以在线方式使用，对新输入分类。不过，诸如矩阵因子分解这样的方法更适合混合离线和在线建模方法：有些因子可以预先以离线方式计算，有些因子可以实时更新，创建更新的结果。其他诸如集群处理这样的非监督方法，也可以对集群中心进行离线计算，对集群节点进行在线作业。这些例子说明：模型训练可以分解为大规模和复杂的全局模型训练，以及轻量级的用户指定模型训练或更新阶段，以在线方式完成。（图2）

对于离线作业（Offline jobs），主要用来运行个性化机器学习算法。这些作业会定期执行，而且不必与结果的请求和展示同步。主要有两种任务这样处理：模型训练和中间与最终结果批量计算。不过，他们也有一些学习算法是以在线增量方式完成的。

这两种任务都需要改善数据，通常是由数据库查询完成。由于这些查询要操作大量数据，以分布式方式完成更方便，因此通过Hadoop或是

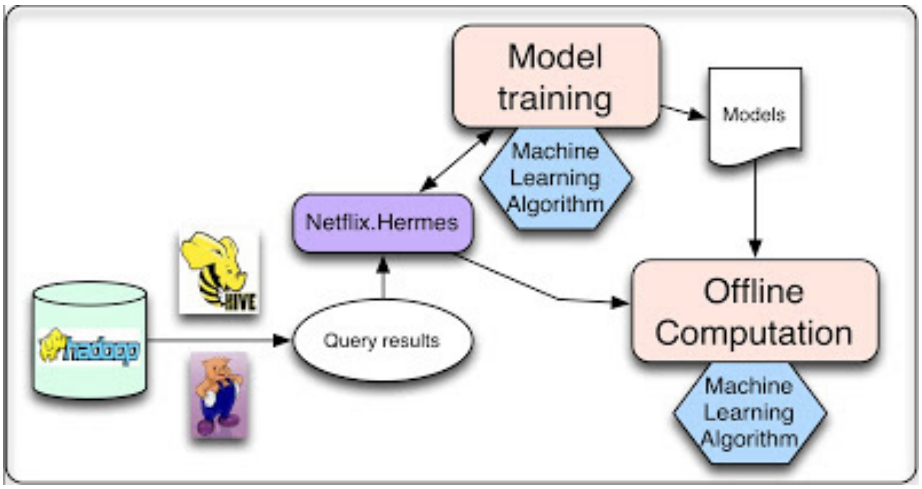

图2

Hive、Pig 作业就是自然而然的事情。一旦查询完成，就需要某种机制发布产生的数据。对于这样的机制，Netflix 有如下需求：

- 可以通知订阅者查询完成；
- 支持不同存储方式（不只是HDFS，还有S3或是Cassandra等）；
- 应该透明处理错误，允许监控和报警。

Netflix 使用内部的工具 Hermes 完成这些功能，它将数据以接近实时的方式交付给订阅者，在某些方面接近 Apache Kafka，但它不是消息 / 事件队列系统。（图 3）

无论是离线还是在线计算，都需要处理三种输入：模型、数据和信号。模型是以离线方式训练完成的参数文件，数据是已完成处理的信息，存在某种数据库中。在 Netflix，信号是指输入到算法中的新鲜信息。这些数据来自实时服务，可用其产生用户相关数据。（图 4）

对于事件和数据分发，Netflix 会从多种设备和应用中收集尽可能多的用户事件，并将其集中起来为算法提供基础数据。他们区分了数据和事件。事件是对时间敏感的信息，需要尽快处理。事件会路由、触发后续行动或流程。而数据需要处理和存储，便于以后使用，延迟不是重要，重要的是信息质量和数量。有些用户事件也会被作为数据处理。

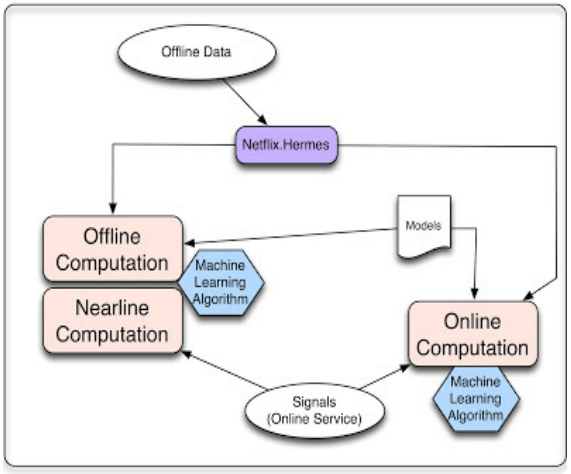

图3

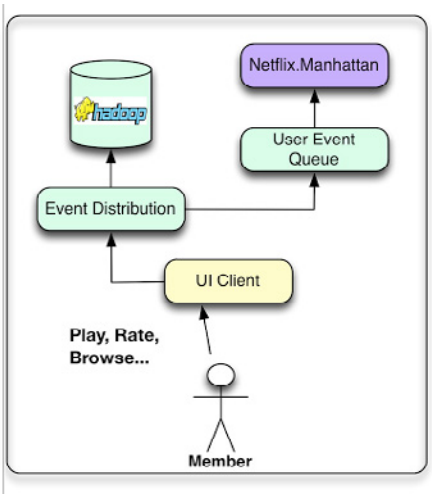

图4

Netflix 使用内部框架 Manhattan 处理接近实时的事件流。该分布式计算系统是推荐算法架构的中心。它类似 Twitter 的 Storm，但是用处不同，而且响应不同的内部需求。数据流主要通过 Chukwa，输入到 Hadoop，进行处理的初步阶段。此后使用 Hermes 作为发布 - 订阅机制。

Netflix 使用 Cassandra、EVCache 和 MySQL 存储离线和中间结果。它们各有利弊。MySQL 存储结构化关系数据，但会面临分布式环境中的扩展性问题。当需要大量写操作时，他们使用 EVCache 更合适。关键问题在于，如何满足查询复杂度、读写延迟、事务一致性等彼此冲突的需求，要对于各种情况到达某个最优点。

在总结中，他们指出，我们需要具备使用复杂机器学习算法的能力，这些算法要可以适应高度复杂性，可以处理大量数据。我们还要能够提供灵活、敏捷创新的架构，新的方法可以很容易在其基础上开发和插入。而且，我们需要我们的推荐结果足够新，能快速响应新的数据和用户行为。找到这些要求之间恰当的平衡并不容易，需要深思熟虑的需求分析，细心的技术选择，战略性的推荐算法分解，最终才能为客户达成最佳的结果。

第 3 章 博客推荐系统

物料准备

众所周知，数据科学的目标是从大数据中创造价值。然而，数据科学也应该满足第二个目标即避免信息过载。一个典型的可以满足这两个目标的系统就是推荐引擎。不仅是像 Amazon 这样的网上商店，同样的还有流媒体服务如 Netflix 公司也遭到了信息过载的困扰。客户可能很容易迷失在其庞大的（百万）产品或电影中。推荐引擎通过呈现给用户可能的选择帮助用户从繁多的产品中缩小选择范围。当然这些推荐引擎可以随机的展现给用户一些可能的选择，但这并没有真正降低信息过载。因此，这些推荐引擎通过应用统计科学来给用户展现更匹配他们期望的结果。例如，一个观看了《Frozen》的 Netflix 用户会从 Pixar 看到类似的儿童电影推荐结果。（图 1）

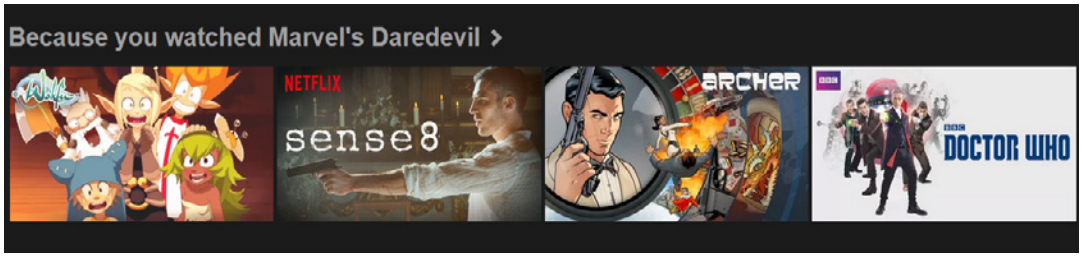

图1

接下来，我们将详细的说明如何为"The Marketing Technologist (TMT)"的读者构建一个推荐引擎。TMT 目前拥有超过 50 个类别的文章，覆盖了众多类别的主题，从数据科学到 ReactJS 编程。浏览完网站上所有的文章是非常耗时的，尤其是在文章的数量一直增长的情况下。同时读者很可能仅仅对他们所熟悉的领域的文章感兴趣。如果一个推荐引擎能够筛选出用户感兴趣的文章，那么它就可以被认为是能从数据中创造价值并避免信息过载的那一类系统。

两种类型的推荐系统

粗略地讲，我们可以将推荐引擎划分为两种类型：协同过滤和基于内容的推荐。就像维基百科说的：协同过滤是一个使用涉及多个代理，观点和数据源之间的协作技术模式来过滤信息或模式的过程。在 TMT 案例中，这意味着在多个读者中找到一些规律。如果一群读者对特定系列中的文章感兴趣，当一个读者开始读这个系列中的一篇文章，那么他有很大的概率会对这个系列中的其他文章感兴趣。因此，可以基于其他用户的阅读行为来给类似的用户进行推荐。

基于内容的推荐引擎是不同的，因为它们是以产品的属性进行推荐。在我们的案例中产品就是 TMT 的文章，属性则是文章中的关键字。如果一个用户在读的文章中包含 Google Analytics 和 Tag Manager 的关键字，那么这个用户很有可能也喜欢包含这些关键字的其他文章。因此，基于内容的推荐引擎会推荐包含这些关键字的文章。值得注意的是最近从“Geek”到“The Marketing Technologist”的升级中，一个非常简单的基于内容过滤推荐方法被集成到 TMT 中。就是在每篇文章的下面有 5 篇其他相关的文章展示给用户供他们继续阅读。推荐的文章就是包含用户在读文章中任意一个标签的最新发表的 5 篇文章。在这个简单的例子中，标签可以当作产品的一个属性。（图 2）

这两类系统各有利弊。基于内容的推荐系统将推荐的文章限定在基于

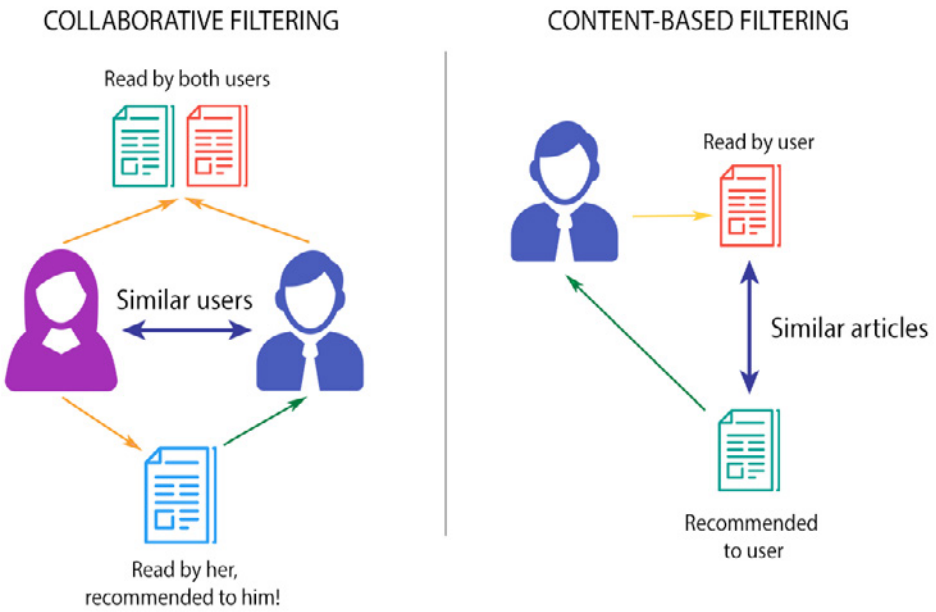

图2 协同过滤和基于内容的推荐背后的原理

一组特征类似的文章中。例如，在给包含 Google Analytics 这样具体特征的文章进行推荐的时候会在其他文章中查找类似的关键字。然而，一个包含类似的分析工具 Snowplow 这样具体特征的文章，却不太可能被推荐出来。实际上用户可能对这两类文章都感兴趣，因为他们都属于分析这个主题。因此，基于内容的推荐系统并不善于发现隐藏的模式。协同过滤推荐系统在寻找隐藏模式这点上是优于基于内容的推荐系统。协同过滤着眼于用户的阅读行为，不专门关注用户阅读文章的内容。所以如果一个用户阅读了关于数据科学和转化率优化 (CRO)，尽管那些 CRO 的文章的内容和数据科学是截然不同的，协同过滤推荐系统也会向数据科学的读者们推荐关于 CRO 的文章。协同过滤的最大的缺点是，它需要大量的历史用户阅读行为数据，以便找到这些模式。基于内容的推荐可以在不给或者少量的历史数据的条件下完成，因此更容易实现。

协同过滤推荐系统的准备前提

在接下来的两篇博客中，我们将会实现一个基于内容推荐系统和基于协同过滤算法的推荐系统，并分析推荐结果。然而，为了能够实现目标，

	Client id	Page	Product Detail Views	Quantity Added To Cart	Unique Purchases
1.	1000527031.1446573579	/caching-http-requests-in-angularjs/	1 (0.02%)	1 (0.03%)	0 (0.00%)
2.	100064687.1446479212	/where-have-my-factories-services-constants-and-values-gone-in-angular-2/	2 (0.03%)	1 (0.03%)	0 (0.00%)
3.	1000997522.1445512444	/4-improvements-in-scrum/	2 (0.03%)	1 (0.03%)	0 (0.00%)
4.	1000997522.1445512444	/5-reasons-why-you-need-a-javascript-style-guide/	2 (0.03%)	1 (0.03%)	1 (0.07%)
5.	1000997522.1445512444	/5-takeaways-from-the-google-analytics-user-conference-in-amsterdam/	2 (0.03%)	1 (0.03%)	1 (0.07%)
6.	1000997522.1445512444	/author/siebe/	2 (0.03%)	1 (0.03%)	0 (0.00%)
7.	1000997522.1445512444	/how npm3 will saves webstorms life/	2 (0.03%)	1 (0.03%)	1 (0.07%)
8.	1000997522.1445512444	/how-to-get-airbnbs-javascript-code-style-working-in-webstorm/	6 (0.10%)	3 (0.09%)	1 (0.07%)
9	1000997522.1445512444	/how-we-keep-our-mobile-device-lab-small-and-simple/	2 (0.03%)	1 (0.03%)	0 (0.00%)
10.	1000997522.1445512444	/setting-up-a-cookie-law-compliant-google-analytics-tracker/	2 (0.03%)	1 (0.03%)	1 (0.07%)
11.	1001103005.1445961038	/setting-up-a-cookie-law-compliant-google-analytics-tracker/	1 (0.02%)	1 (0.03%)	1 (0.07%)

图3 谷歌分析中一个自定义的用户层面的阅读行为报表示例

我们首先需要准备一些前提物料。对于协同过滤推荐系统这意味着我们需要实现得到用户阅读文章行为的方法。我们的同事 Erik Driessen 已经实现了通过使用谷歌分析并且分配客户端 ID 来记录用户阅读行为的方法。Simo Ahava 在他的博客《在谷歌分析中存储客户端 ID》解释了如何有效的实现它。

此外，由于 Erik 已经在文章的内容上增加了 Enhanced Ecommerce 插件，这样我们就可以追踪到用户是否完整的阅读了这篇文章。最后，在谷歌分析中可以创建自定义报表，展示客户端 ID 和他们已经阅读过的文章。

需要注意的是，目前跨设备的解决方案尚未实现。因此，如果用户使用不同的设备继续阅读文章或着中间清除了 cookie，后续的行为将不能和他前期的阅读行为联系起来。（图 3）

基于内容的推荐系统的准备前提

在基于内容的推荐中，显然我们会需要所有 TMT 文章内容。有多种实现的方法。其中的一个将是直接从数据库中提取这些文章的文本。然而，因为我们是 Geek，通过编写一个 Python 脚本，可以自动检索文章和相应

的指标如作者和类别，这将是一件更酷的事情。因此，我们创建了从 TMT 网站复制文章的 Python 脚本，具体分为两个步骤。步骤 1 和步骤 2 的代码在这里（注：英文原文并未给出链接）。

步骤 1： 创建一个 TMT 全量文章的列表 .

可以使用 Python 的 urllib2 库加载网页源代码。用 urllib2.urlopen("http://www.themarketingtechnologist.co") 这个命令可以加载我们自己 TMT 博客的前端页面的源代码。这个首页总是展现最新发表的 10 篇文章。使用 BeautifulSoup 库，我们可以很容易通过在 DOM 中搜索并提取所有带 class ="post" 的 article 元素，并将它们存储在一个 Pandas dataframe 中。此外，在每个这样的元素内部，我们还可以通过搜索相应的 DOM 元素来提取作者和标签等元素。

因为只有十篇最新的博客文章在首页显示，为了获取更多博客文章，我们还需要检查在页面底部是否有 Older posts > 按钮。同样的，可以通过搜索相应的 DOM 元素，如 class="older-posts" 来完成。在更早博客文章区块中可以通过使用 get 函数从 href 属性值中提取到下一个页面的链接。在每个页面上我们重复上述过程。最后，我们将得到一个包含所有文章的名称、标签、作者和指向文章内容链接的数据帧。

步骤 2：检索每篇文章内容

在步骤 1 中，我们保存了一个到每一篇文章的链接，所以我们可以下载每篇文章的全部内容。但是存在一个特殊的问题，即每篇博客文章的内容是通过 JavaScript 加载的。因此，如果我们使用的 urllib2 加载文章的静态资源，我们将无法得到文章的内容。为了执行 JavaScript 代码来加载文章内容，我们实际需要通过 Web 浏览器进行渲染出文章的内容。幸运的是，这可以通过使用流行的 Selenium 库来达到目的。使用 Python 的几行代码并结合 Selenium 库可以打开 Firefox 浏览器，定位到相应的 URL 并渲染页面。然后就可以在 DOM 中搜索我们想要的信息，例如博客文章的

内容。

需要注意的是，因为使用 Selenium 执行所有的 JavaScript 代码，这也意味着谷歌分析的代码也会被执行。因此，明智的做法是采取措施防止产生脏数据。例如，在 GA 过滤器列表加入您的 IP 地址或者通过谷歌安装 Google Analytics Opt-out Addon 插件。还要注意的是，实际上并不需要在 Firefox 浏览器中把每个渲染出来页面呈现在我们面前。你也可以使用一个轻量级的驱动程序，如 PhantomJS 这使得可以在后台渲染页面而减少视觉上额外的开销。

这就是截至到现在建立推荐系统需要的准备前提。在接下来的几个月中，我们需要收集协同过滤模型中用到的用户层面的阅读行为。因此，在下面的博客文章中，我们先开始创建一个基于内容的推荐系统，并分析其结果。复杂度限制更少，因为它以批量方式完成，没有很强的时间要求。

基于内容相似性的推荐

前面我们介绍了推荐系统的优点，大致可以把推荐系统分为两种类型：基于内容的推荐系统和协同过滤推荐系统。接下来，我们描述度量文章相似性的方法进而着手于基于内容的推荐系统中的第一个步骤。

基于内容的推荐系统背后的概念

目标是为我们的读者推荐 TMT 上的其他文章。我们假设读者已经完整的看过一篇喜欢的文章，并希望阅读更多类似的文章。因此，基于用户的历史阅读行为，我们想要打造一个基于内容的、可以推荐新的类似文章的推荐系统。

为了得到准确和有用的推荐结果，我们希望用一个数学量化的方法来找到最好的推荐结果。否则，我们的网站会减少对读者的吸引力，进而他们会离开 TMT。因此，我们希望找到那些彼此相似的文章，从而它们就是“彼此相近”的。也就是说，这些文章之间的“距离”是近的，一个较小的距

离意味着更多的相似性。（图 4）

那么，这个距离的概念究竟是什么？假设我们可以将任意一篇 TMT 的文章映射为 2 维空间中的一个点。上面的图提供了一个将 76 篇 TMT 文章映射在 2 维空间中的例子。此外，我们假设两点越接近它们彼此就越相似。因此，如果把用户在读的文章当作 2 维空间中的一个点，则在这个空间中离这个点距离越近的点代表的那些文章可以被视为好的推荐结果，因为它们是相似的。这些点之间的距离有多远可以使用欧几里德距离公式来计算。在 2 维空间这个距离的公式可以简单地归纳为勾股定理。需要注意的是，这个距离公式也适用于更高维度的空间，例如 100 维空间（尽管超过 3 维空间的时候我们已经很难想象了）。

余弦相似性是另外一种计算两点的相似性的公式。想象一下，为每篇

$$\text{Euclidean Distance} = \sqrt{(x_1 - y_1)^2 + \ldots + (x_N - y_N)^2}$$

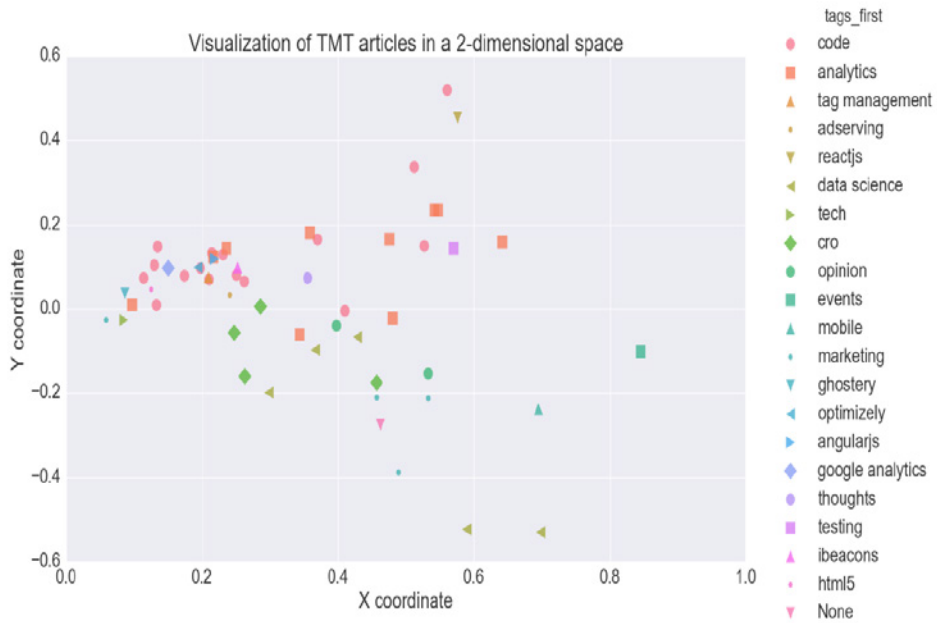

图4 一篇TMT文章在2维空间可视化的例子

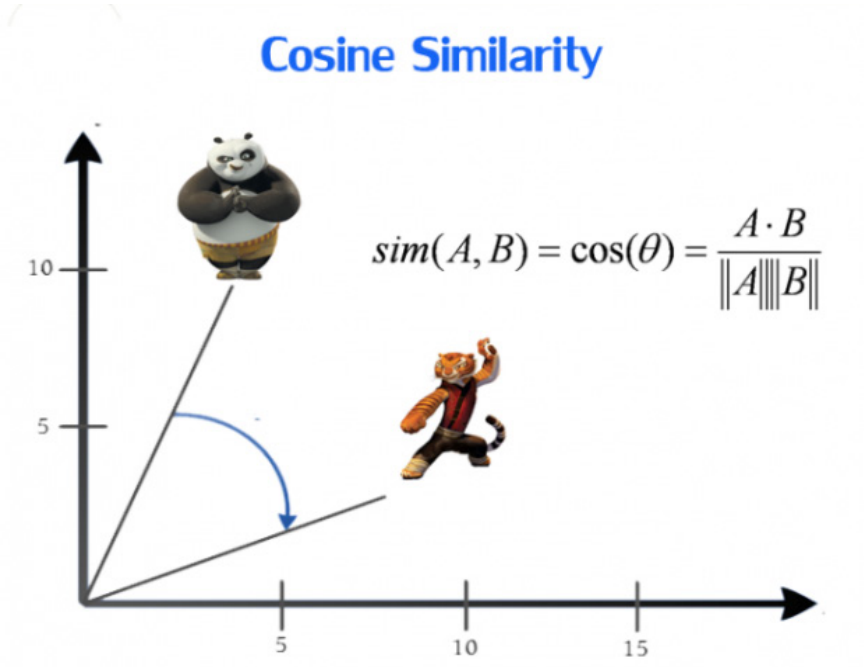

图5 余弦相似性

文章分配一个它倾向的方向。余弦相似函数使用的是两篇文章指向的方向差，即两篇文章方向之间的角度差。例如，在 2 维空间中，一篇文章方向是正北向和另外一篇文章是正西向，则他们在方向上的角度差是 -90 度。把这种角度差归一化在区间 [-1,1]，其中 1 表示的是相同的方向，因此是完美的相似，-1 表示完全相反的方向，因此是完全没有相似性的可言的。（图 5）

我们使用余弦相似性量度 TMT 文章的相似度是因为文章之间的方向比文章之间的精确距离更重要。此外，我们尝试了两个度量指标（欧氏距离和余弦相似性）并且余弦相似性度量指标表现的更好。

两个新的挑战

上面我们介绍了度量文章之间相似性的方法的概念。然而，现在两个新的挑战出现了：我们如何把每篇文章映射成 2 维空间的点？

空间中的两点间的距离提供了有关文章的相似性的指标，那么我们如何绘制这些代表文章的点之间的距离？

一篇文章可通过在 2 维空间中分配给它的坐标来绘制，例如一个 x 和 y 坐标。这意味着首先需要把我们的文章转换为数字格式，然后将其归约为两个值，例如 x 和 y 坐标。因此，我们首先要阐述一种通过应用特征提取算法来科学的量化 TMT 文章中文本的方法。

需要注意的是，我们将在下面讨论的处理文本的特征提取方法是专门为处理 TMT 中的文章而设计的。你可以想象一下，如果你正在构建一个针对电话的基于内容的推荐的系统，你可能需要一种不同的算法来将电话的属性（内容）转换为数字格式。

将TMT文章变换为数字格式

TMT 的文章是由大量的单词组成的短语和标点符号组成，需要被变换为数字向量，这个变换的过程中不允许损失文章的内容。我们希望最好是一个固定大小的向量。为什么我们想要固定大小？回想上面提到的 2 维空间的例子。将 2 维空间中的点和 100 维空间中的点进行比较是很荒谬的，对吧？此外，如果我们讨论两篇文章某些方面的相似性就需要用相同的方式来描述它们。

为了得到固定大小的向量我们将创建一些特征，例如那些文章可以测量的属性。在文本分析领域中，特征往往指的是单词、短语、数字或符号。所有的文章都可以被一个相同的特征集合来量化并表示，进而所有的文章就被变换为固定大小的数字向量。将文本变换为数字向量的整个过程称为特征提取，通常分为 3 个步骤：分词，计数和加权。（图 6）

步骤 1：分词

在特征数字化的第一步是分词。在文本分析中，分词被认为是从大量的文本中得到有意义的词汇单元。例如，物理速度可以用米 / 秒来表示。在文本分析中，文本的长字符串可以用词汇单元来表示。这些词汇单元通常对应单词。因此，获取句子 I am happy, because the sun shines 词汇单元的一种简单的分词方法就是通过空格进行分割。用这种分割方法

图6 文本向量化的例子

得到 7 个词汇单元，例如，I， am， happy,， because， the， sun，shines。分词之后，原始的句子就可以用这些词汇单元来表达了。

这种简单的分词方法引入了新的问题。

第一，这种方法不会过滤掉任何标点符号，从而导致了 happy, 这个词汇单元是以逗号结尾的。这也就表明 happy, 和 happy 是两个不同的词汇单元，尽管他们实际上代表的是同一个单词。因此，我们需要过滤掉所有的标点符号，因为标点符号几乎和我们的文章中单词的含义是不相关的。需要注意的是，在其他场景中，标点符号可能和单词的含义是相关的。例如，在分析 Twitter 中的消息时，标点符号也是很重要的，因为它们通常被用来创建可以表达很多情感的笑脸。笑脸这个例子强调了每种数据源需要有自己的特征提取方法的事实。

第二，利用简单的分词方法，可能会获得的分词结果包含 works 和 working。然而这些都是同一个单词的不同形式，例如 to work。当一个单词是另一个单词的复数形式时，同样也是上面的这种情况。在我们的基于内容的推荐系统中，我们假设这些单词不同的形式代表相同的含义。因此，这些单词可以归约到它们的词干形式并当作同一个词汇。为了达成这个目标，我们需要一种词干提取算法，可以把每个单词变换成其词干形式。需要注意的是这种词干提取算法每种语言都是专用的。幸运的是，已经有

一些可用的免费程序库，例如 NLTK 这个库可以帮我们完成这个工作。

第三，通常文本分析的存在的问题是如何处理单词的组合或者单词的否定语意。例如，如果仅仅用单独的单词 Google 和 Analytics 并不能表明我们正在谈论的是 Google Analytics 这个产品。因此，我们需要创建两个或三个连续单词的短语，分别称为二元的 bi-gram 和三元的 tri-gram。I like big data 这个句子，可以被转换为 I， like， big，data，I like， like big， big data， I like big 和 like big data 这些词组。

需要注意的是，这种分词方法并没有把单词所在的位置以及顺序考虑进来。例如，分词之后就不能确定单词 big 在原始的句子或文章出现的位置了。同时，单词自身也没有携带它前面和后面单词的任何信息。因此，我们在分词的过程中丢失了原始句子的某些信息。关键点是在保存一系列分词的同时也采集尽可能多的句子的原始信息。用我们的分词算法将会丢失原始句子的结构信息。有其他会把单词在文章中出现的位置和顺序考虑进来的分词算法。例如，Part-Of-Speech （POS）方法还增加了额外的标注信息，例如词性，如词汇是否是一个动词，形容词，名词或专有名词。然而，我们假设词性标注不会大大增加我们推荐引擎的推荐效果，因为词语在句子中的顺序在推荐的时候不是很重要。

步骤 2：词频统计

在第二步骤中，需要对每篇文章中的每个词汇单元的频率进行统计。这些词频数据将用于下一步骤，为文章中的词汇加权。此外这些词汇的计数信息将被用于稍后执行初步的特征选择，例如用来减少特征的维度。注意，文本分析的典型特性是，大多数的词汇仅在几篇文章中使用。因此，大多数的词汇在一篇文章中出现的频率为零。

步骤 3：加权

在最后步骤中，使用前面步骤得到的词汇和词频把所有文章转换为数字格式。把每篇文章编码成一个数字向量，向量的元素代表的就是步骤 1

得到的词汇单元。此外，词汇加权过程使用从第 2 步得到的词频。当给词汇赋完权重之后，它不在被认为是词汇而是被当作一个特征。因此，一个特征代表一个词汇，文章特征的的值是由加权方法确定的权重。

有多种可选的特征加权方法：

最基础的加权方法就是特征频率 (FF).FF 加权方法简单的使用文章中每个词汇单元出现的频率作为它的权重。例如，给定词汇的集合 {mad,happy}，句子 I am not mad, but happy, very happy，加权得到的结果向量就是 [1,2].

Feature Presence (FP) 是一种类似的基础加权方法。在 FP 算法中，一个词汇被简单的赋予一个二元变量，1 代表这个词汇在这篇文章中出现，0 则相反。前面的例句用 FP 算法得到的结果向量是 [1,1]，因为这两个词汇均在这个句子中出现了。使用 FP 加权算法一个额外的优势就是我们将会得到一个二元数据集，不会遇到扩展性的问题。扩展性问题可能在算法中遇到，例如在复杂计算中像计算特征值或 Hessian 矩阵时。

一种更加复杂的特征加权算法就是词频和逆文档频率 (TF-IDF) 加权算法。这个方法使用两种类型的评分，例如词频评分和逆文档频率评分。词频评分就是计算这个词汇在这篇文章中出现的频率。逆文档频率评分可以由总的文章数目除以包含该词语文章的数目，再将得到的商取对数得到。当这两个评分相乘时，如果得到的值比较大，表示该词语在一小部分文章里频繁地出现很多次，如果值较小则表示该词语在很多文章里都有出现。

在我们的基于内容的推荐系统中，我们将使用 FP 加权算法，因为这个方法非常快并且在推荐效果上不会比其他加权方法差。此外，用它得到的稀疏矩阵，在计算的时候有额外的优势。

降维

应用上述特征推荐方法后，我们会得到一个特征列表，用这些特征可以把每篇 TMT 文章数字化。我们可以在这个高维空间中计算出任意两篇文

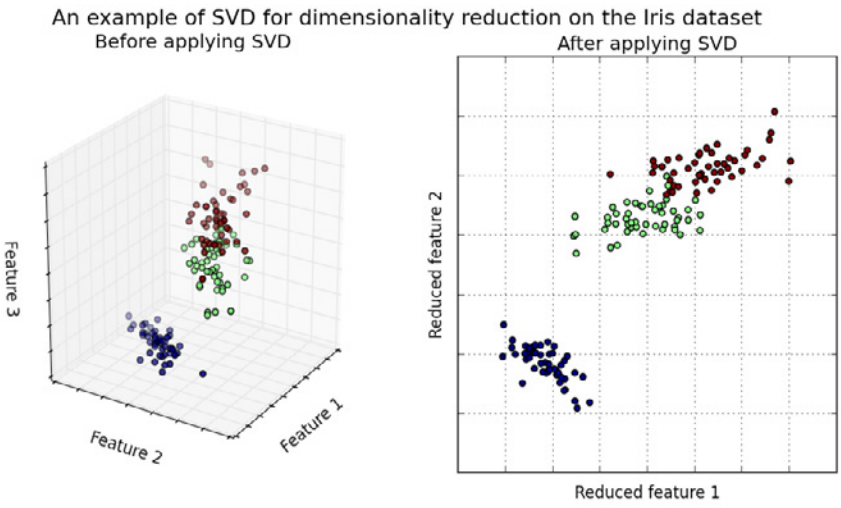

图7 一个用SVD方法对Iris数据降维的例子

章间的相似度，但我们不倾向于这么做。因为有些特征不太可能表达文章之间的相似度，就像 is 和 a，可从特征集合中移除掉，进而显著减少特征的数量，并提高的推荐结果的质量。此外，更小的数据集提高了推荐系统的速度。

文档频率 (DF) 选择是用于降维最简单的特征选择方法，是很多文本分析问题的必需。我们首先删除最常见的英文停用词，例如 the，a，is 等在判断文章之间相似度时并不能提供太多信息的单词。做完这些，那些有着很高和很低的文档频率的特征也会被从特征集合中移除掉，因为这些特征也不太可能在判别文章相似度时起太大的作用。

回想一下，在本文的开头，我们可以想象在 2 维空间中的文章。然而，应用 DF 后我们仍处于一个非常高维的空间。因此，我们打算使用一种算法将高维空间降到 2 维空间，这让我们可以巧妙地将文章可视化。此外，在 2 维空间中很容易理解推荐系统的原理，因为距离的概念是很直观的。用来把我们带回了 2 维空间的算法是奇异值分解（SVD）。另外一种不同的算法是主成分分析（PCA）。我们在这篇文章中不会全面的介绍这些算法是如何工作的。总之，两者的本质都是找到最有意义的基，用这组基我们可以重造原始数据集和捕获尽可能多的原始方差。幸运的是，scikit-

learn 已经包含了内置版本的 SVD 和 PCA 算法，因此我们可以很容易使用。

还有很多用来降维的方法，如信息增益和 X 平方准则或随机映射，为了简单起见，我们坚持使用 IDF 特征选择方法和 PCA 降维方法。（图 7）

用 SVD 将从 3 维降到 2 维，并且不会丢失太多的信息。

标记推荐结果

应用所有上述步骤之后，所有的 TMT 文章被归约成一个 2 维空间的坐标。对于任意两篇文章，现在可以计算两个坐标之间的距离。我们现在唯一需要的一个函数，这个函数可以将当前的文章作为输入，返回与这篇文章固定数量的距离最小的 TMT 文章！使用欧氏距离公式这个函数是很容易实现的。

让我们构造一些场景来测试我们的基于内容的推荐系统。假设我们是刚刚读完 Caching $http requests in AngularJS 这篇文章的读者，基于内容的推荐系统提供了以下 TMT 文章推荐结果供我们继续：Angular 2: Where have my factories, services, constants and values gone? 听起来很合理，对吧？

总结

我们将用几句话总结我们基于内容的推荐系统的第一步。

在最终的推荐系统，我们使用 SVD 算法将维度减少到 30 的而不是 2。这样做是因为当我们将其降低到一个 2 维空间，很多特征的信息将会丢失。因此，文章的相似性量度指标也是不太可信的。我们使用 2 维空间只是做可视化的。我们还对 TMT 文章的标题和标签应用了特征提取算法。这显著的改善了推荐结果的质量。

我们的推荐系统的参数是直观选择的，并没有做优化。这将在以后的文章中阐释！我们希望您从这篇文章学到了一些东西！

第 4 章 Spotify 每周歌曲推荐算法解析

Spotify 是全球最大的正版流媒体音乐服务平台，2008 年 10 月在瑞典首都斯德哥尔摩正式上线。Spotify 提供的服务分为免费和付费两种，免费用户在使用 Spotify 的服务时将被插播一定的广告。而付费用户则没有广告，且可以拥有更好的音质，在移动设备上使用时也可以拥有所有的功能。截止到 2015 年 6 月，Spotify 已经拥有超过 7500 万的用户，其中 1500 万为付费用户。

每个周一，7500 万 Spotify 用户都会收到了一卷新的混音带，这是由 Spotify 的 Discover Weekly 通过算法每周向用户专门推荐的播放列表，里面包含了 30 首歌曲，它带来了非常好的用户体验，感觉像是一个爱好音乐的朋友专门送给用户的礼物。

自动音乐推荐已经不是什么新鲜事，但 Spotify 似乎可以让用户对个性化推荐的播放列表同时感到新鲜和熟悉。一些潜在的竞争对手，如 Pandora、Google 和 Apple，很大程度上都可以为用户提供深不见底的音乐目录，但是 Spotify 通过采用不同的方法可以为每个用户挑选出用户喜欢的音乐。

Spotify 的这项服务得到了用户广泛欢迎。根据 Spotify 的消息，自

从 6 月份 Discover Weekly 悄然推出之后，播放列表中的歌曲被播放过 17 亿次。如果由于什么原因，推荐的歌曲列表推迟发布了，往往会让很多用户感到非常低落，感觉到生活中缺少一种大的乐趣。

“相比过去，我们现在有更多的技术保证，即使你是世界上最小众、最奇怪的音乐家，做着世界上只有 20 个人懂得欣赏的音乐，我们也能找到那 20 个人，并在你们之间建立联系”，在 Spotify 负责监视这项服务的工程师 Matthew Ogle 表示，“Discovery Weekly 正用一个非常引人注目的新方式来做到这一点，而在以前从来没有人做到过”。

如果你是一个 Spotify 用户，在你登陆之后，将可以看到 Spotify 这周为你挑选的歌曲（如果你不是 Spotify 用户，或者没有登录，你会看到自己最近的播放列表）。

下面将对这个过程进行简单的介绍。

收集所有用户的播放列表

Discover Weekly 的主要组成部分来自于所有用户。 Spotify 首先查看用户创建的 20 亿个播放列表，每一个列表都反映了用户的偏好。这些用户对于歌曲的选择和分组形成了 Discover Weekly 推荐的核心。

发现用户自己的收听口味

但 Discover Weekly 为用户推荐的播放列表要更为复杂。 Spotify 针对每个用户的音乐品味，为每个用户创建了一个 profile，按照艺术家和音乐流派的不同进行分类，不仅仅是划分为常见的“摇滚”和“说唱”，而是更细粒度的，如“合成器流行乐”和“南方灵魂乐”等等，这些可以通过 Echo Nest 的技术进行分析，Echo Nest 是 Spotify 在 2014 年收购的一家音乐分析公司，它通过机器读取音乐网站从而了解到新的专辑，并对音乐进行进一步分析。

用户可能不知道什么是“流行 pop”或者其他类型的流派，但 Spotify 根据用户自己的 Spotify 收听数据，会告诉用户你原来是一个流行 pop 音乐的粉丝。（图 1）

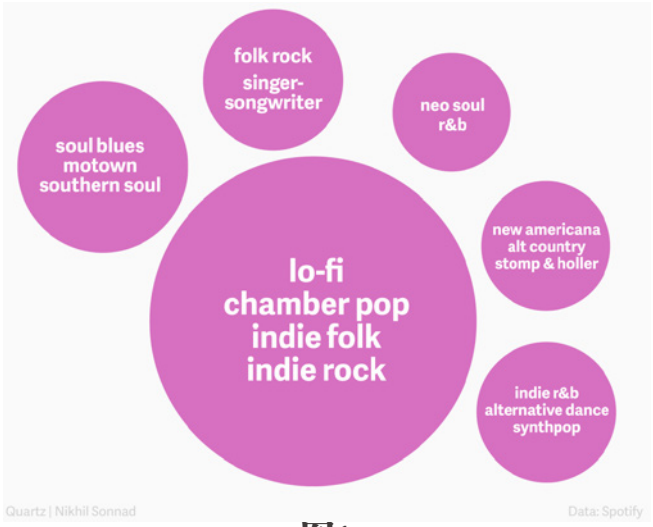

图1

通过算法进行集成

通过 Spotify 的算法可以建立 20 亿个播放列表的数据与用户个人口味之间的联系。

Spotify 的工程师在今年早些时候的一个报告中分享了许多的技术细节。他们的方法包括：协同过滤，例如在亚马逊中经常见到的功能“客户谁买了这个 item 也会买 .....”，以及自然语言处理，Echo Nest 是如何理解音乐博客和播放列表的标题的。公司采用开源软件 Kafka 来对数据实施实时管理。（图 2）

寻找适合用户的音乐

Spotify 采用了深度学习技术，深度学习是可用于海量数据分析的机器学习技术，能够大大提高 Discover Weekly 的推荐水平。这项工作由 Sander Dieleman 构建，Dieleman 曾是 Spotify 的实习生，现在是谷歌的 AI 子公司 DeepMind 的研究科学家。

Discover Weekly 的首席工程师 Edward Newett 表示，“我们已经用深度学习和神经网络试验了不同的方法，并且这是 Discover Weekly 中最重要的功能之一”。

用深度学习预测用户收听兴趣

关于 Spotify 采用深度学习进行音乐推荐的方法，Dieleman 和他的

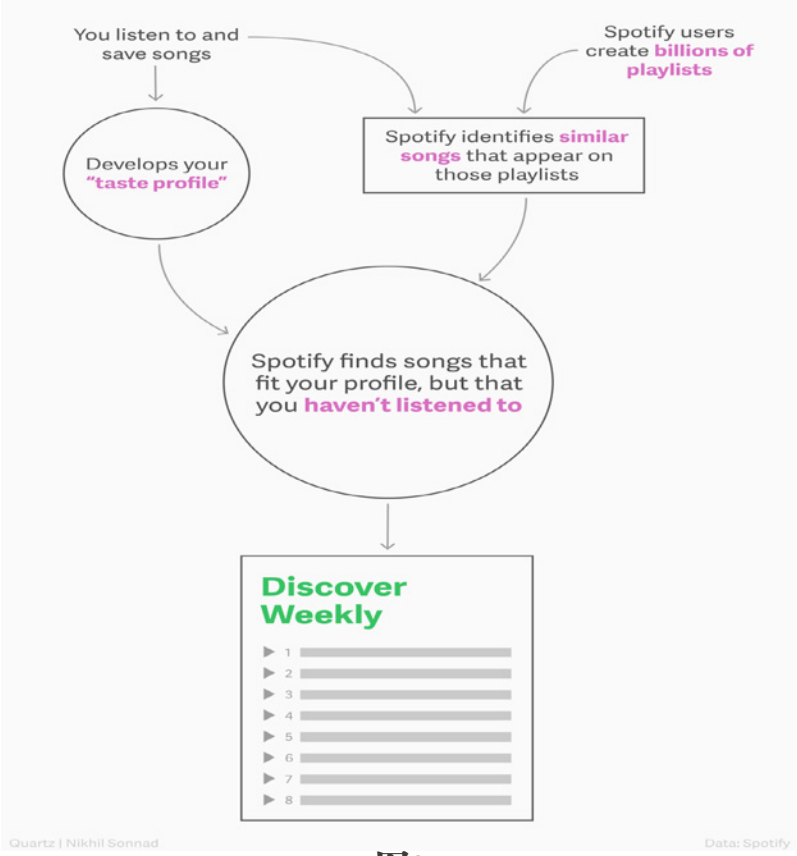

图2

同事 Aäron van den Oord 一起在 NIPS2013 上发表了一篇关于这个推荐系统的文章，题为“Deep content-based music recommendation”。他们试图使用音频信号，通过训练一个回归模型来预测歌曲的隐特征表示，从而解决用户收听偏好预测问题。通过这种方式，即使没有用户行为数据可用，也可以在一个协同过滤空间中预测一首歌曲的特征表示。（正如你可能从本文的标题推断的，解决这个问题的回归模型是一个深度神经网络。）

这种方法的基本想法是，通过协同过滤模型将用户和歌曲投影到一个共享的低维隐空间中。一首歌曲在空间中的位置编码了所有反映用户偏好的信息。如果两首歌曲在空间中非常靠近，它们可能非常相似。如果一首歌曲与另一个用户非常接近，它对于用户来说可能是个好的推荐（只要他们以前没有听过）。通过分析音频预测一首歌曲在空间中的位置，就能够选择合适的歌曲推荐给一个正确的听众，而不依赖于用户的历史行为数据。

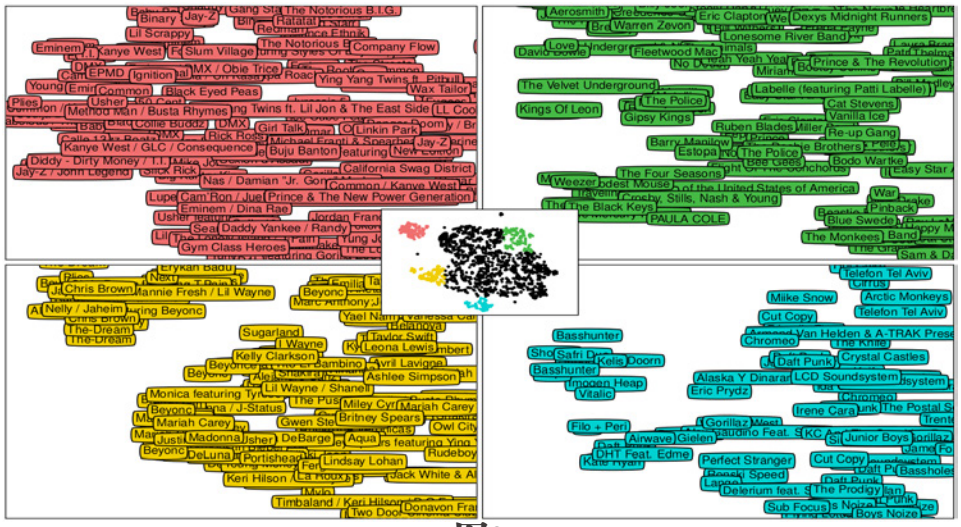

图3

通过使用 t-SNE 算法可以将模型的预测结果投影到一个低至二维的隐空间中，这个过程的形式化如下图。正如图中看到的结果，相似的歌曲会聚类在一起。Rap 音乐主要集中在左上角，然而电子乐歌手主要聚集在底部。（图 3）

Dieleman 表示选择处理这个问题的主要原因是因为他相信音乐音频信号的推荐一个极其复杂的问题，需要很多层次的抽象才有可能实现。他希望网络的连续层次能够逐渐学习到更复杂和不变的特征，以便更好地用于图像分类任务。

Spotify 在它们的推荐流水线中已经使用了一堆不同的信息源和算法，Dieleman 表示，他的工作的最直接的应用是将它作为一种额外的信息。然而，它也可以被其他算法用来过滤掉推荐中的离群值。例如，协同过滤算法的推荐结果中往往会包括开场曲目，尾奏曲目，翻唱歌曲和混音作品。这些都可以通过使用基于音频的方法被有效过滤掉。

如果你对深度学习、特征学习以及它们在音乐推荐中的应用感兴趣，可以查看得 Dieleman 的 research page，上面有 Dieleman 在这个领域中所做的一些主要工作。如果你还对 Spotify 的音乐推荐方法有兴趣，还可以查阅 Erik Bernhardsson 博客。

有点神奇的个性化

Adam Pasick 说到，正如许多 Spotify Discover 用户所表示的，推荐的结果经常让人感觉很神奇，Spotify 是如何将上世纪 90 年代的摇滚乐队 Dinosaur Jr. landed 的歌曲加入到播放列表中的？

“我最喜欢的事情之一就是 Discover Weekly 的怪异”，Ogle 表示。“我们可以建立这样庞大的系统，它提取了数以百万计的用户喜好，自动为用户进行音乐推荐，不需要专门通过人力来解决”。

每周的用户播放列表都不相同，大概反映了用户变化的移动音乐喜好。在一个典型的有 30 首歌曲的 Discover Weekly 播放列表中，用户会找到大约 15 首歌是很喜欢的，10 首歌觉得很普通，4 首歌是完全不会听，1 首是非常痴迷的。

Pasick 说到，最近有一天，他到了他最喜欢的一间咖啡馆，音响中传来了非常熟悉的声音。录音带上的许多曲目都是 Spotify 通过算法为 Pasick 推荐的，但咖啡馆的音乐都来自于一名乐队乐手 Homero 的 Discover Weekly 播放列表，但 Homero 不在自己的乐队中的时候，会兼职作为一名咖啡师。

Pasick 很好奇，Homero 和他是怎么得到同样的歌曲列表的？是从 20 亿个播放列表中，是通过复杂的算法为他俩过滤出相同的结果的吗？Spotify 的人工编辑会干预这个过程吗？

有人会说，“哦，我们三个人从 Discover Weekly 得到了相同的推荐列表，这个列表是有人把它放那的吗？”。 Ogle 表示，“答案是肯定的，当其他的 Spotify 用户在建立自己的播放列表，意味着全世界有很多人由于跟他有相同的音乐文化，就会有相同的播放列表”。

他说，尽管艺术家和他们的唱片公司一再要求，Spotify 还是没有使用特定歌曲去刻意建立一个个的播放列表。

深入别人脑海的旅行

Pasick 说到，当你在听其他人的 Discover Weekly 播放列表的时候

会感觉有些神奇，就像他那天在咖啡馆遇到的情况一样，这种感觉就好像在哄骗其他的 Spotify 用户与人分享他们的播放列表。这有点像进行了一个短暂的旅行，既像地理上的，又像是在别人脑海中的。

“当我年轻的时候，经常会有这样的事情，回家后打开 CD 机，通过所听音乐，发现自己真实的存在” ，工程师 Newett 说到，“但现在，你可能会意识到，你曾经以为你是独自一个人在世界上，但现在会感觉不一样，你会发现世界上存在那么一些，至少在音乐爱好上跟你一样”。

使用Discover Weekly 的专业技巧

目前，Discover Weekly 并不完美。Pasick 说，他的播放列表中通常包含一两首歌曲是他绝对喜欢的（Dengue Fever 的“Tiger Phone Card”，以及 Mar Superstar 的“Lady You Shot Me”），有一半是非常喜欢的（Gregory Porter 的“1960 What?”），还有几个讨厌的歌手（为什么我总是收到这么多 Neil Diamond 的歌曲？）。

为了确保从 Discovery Weekly 中获得最喜欢的歌曲，用户可以如何微调他们的结果呢？除了来自其它重要的 Spotify 用户的一些提示，Spotify 还给出一些建议，从非常简单到非常复杂的都有，包括：

添加你喜欢的歌曲到播放列表或 Spotify 库。“如果你保存一首歌曲到你的播放列表或到 Spotify 库，然后开始定期地调整它，它会真正影响 Spotify 对你的理解”，Ogle 表示。

跳过你不喜欢的歌曲。如果一首歌曲在前 30 秒内就被用户 pass 掉了，Discover Weekly 算法将给这首歌曲和相应的歌手一个“thumbs-down”。

听听新的歌手和他们的音乐。“如果我们给你推荐一些东西，你通过点击歌手，开始查看他们的唱片，并同时加入到播放列表中”，Ogle 说。“你在 Discover Weekly 之外浏览得越多，就越有可能影响我们为你做的推荐”。

耐心一点。该算法在设计中会忽略新的收听行为中一些迅速地、突然

出现的峰值，因为许多人分享他们的 Spotify 登录，因此任何新的收听活动可能不会导致你的播放列表的立刻变化。

如果你不想 Spotify 注意到你的行为，可以使用“私密模式”。Newett 表示，“也许你不想让别人知道你的偏好，我们会按照这种“私密模式”，而不跟踪你的曲目播放”。Spotify 会忽视你在 Discovery Weekly 中收听的歌曲。“同时，我们还有一些担忧，比如，如果我只听 Discovery Weekly，会不会出现蛇吞尾巴的现象？”，Newett 说。

将一些歌曲风格过滤掉。Ogle 说到，Spotify 确实对哪些用户可能想要哪些歌曲做了一些编辑决策，因此有年幼小孩的父母不会从 The Wiggles 获得一百万首歌曲，圣诞歌曲在 12 月 25 日之后将大大消失，在大多数情况下，我们试图为它们设置一个护栏，但同时也不是绝对的，因为 Spotify 保留了用户作为人的自主权利，我们应该尽量尊重。

音乐心灵感应的实验。为了更加深入了解 Spotify 的推荐，最好的方式是收听其他人的 Discovery Weekly 播放列表。有一个很有品位的朋友吗？问他分享一个他的播放列表的链接。默认情况下是专有的，但能够被用户分享。

用 IFTTT 保存 weekly 播放列表。Discover Weekly 的一个缺点是，播放列表每周一会被擦拭干净。你可以手动地将歌曲保存到另一个播放列表中，也可以使用免费的 IFTTT 服务自动保存你的 weekly 播放列表到一个单独的存档播放列表。

使用 Spotify 的“广播”功能。如果你想听到一些新的声音，并且绝对不能等到周一，右击 Discovery Weekly，选择“Start Playlist Radio”，Discovery Weekly 服务将按照你的 weekly 播放列表，尽最大努力提供一个与其相似的无穷列表。

第 5 章 达观数据个性化推荐系统实践

在 DT(data technology) 时代，人们的日常生活已经和各种各样的数据密不可分，例如在网络购物、在线视频、在线音乐、新闻门户等都在产生海量的数据。海量的数据产生也带来了信息过载和选择障碍的困扰，每个用户的时间和精力是有限的，怎样帮助用户进行信息的过滤和选择，在 DT 时代是非常有价值的。

面对“信息过载”，个性化推荐根据用户的历史行为数据进行深层兴趣点挖掘，将用户最感兴趣的物品推荐给用户，从而做到千人千面，不仅满足了用户本质的信息诉求，也最大化了企业的自身利益，所以个性化推荐蕴含着无限商机。

美国著名视频网站 Netflix 曾举办过全球的推荐系统比赛，悬赏 100 万美元，希望参赛选手能将其推荐算法的预测准确度提升至少 10%。美国最大的视频网站 YouTube 曾做过实验比较个性化推荐和热门视频的点击率，结果显示个性化推荐的点击率是后者的两倍。号称“推荐系统之王”的电子商务网站亚马逊曾宣称，亚马逊有 20% ～ 30% 的销售来自于推荐系统。其最大优势就在于个性化推荐系统，该系统让每个用户都能有一个属于自己的在线商店，并且在商店中能招到自己最感兴趣的商品。

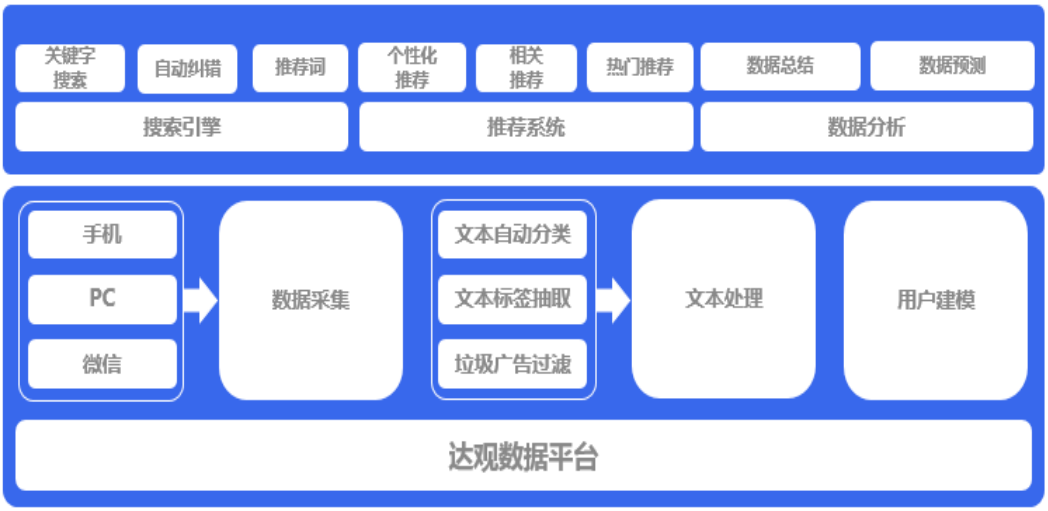

图 1

达观数据（图 1）是专注于企业大数据应用服务的高科技创业公司，致力于为新媒体、电商、视频、金融、企业等提供高质量的大数据挖掘技术服务，通过推荐系统等数据技术服务快速提升企业的经营业绩。达观数据拥有雄厚的研发推荐系统的技术积累，曾在 ACM、CIKM、KDD、Hackathon 等国际竞赛的获奖，在内容推荐，文本挖掘、广告系统等方面申请有超过三十项国家发明专利。本文从数据处理、用户行为建模到个性化推荐，分享达观数据在个性化推荐系统方面积累的一些经验。

1. 数据收集及预处理

推荐系统的本质其实就是通过一定的方式将用户和喜欢的物品联系起来。物品和用户自身拥有众多属性信息进行标识。

物品表示推荐系统的客体，在不同应用场景下，物品指代不同的待推荐事物。比如，在书籍推荐中，物品表示书籍；在电商推荐中，物品表示商品；在电影推荐中，物品表示电影；在社交网络推荐中，物品表示人。商品的多种属性标识不仅标识了是什么，也反应了和其它商品的关系，同时也为量化用户需求的兴趣点提供了依据。用户表示推荐系统的主体，自身属性包括人口统计学信息以及从用户行为数据中挖掘分析得到的偏好等。

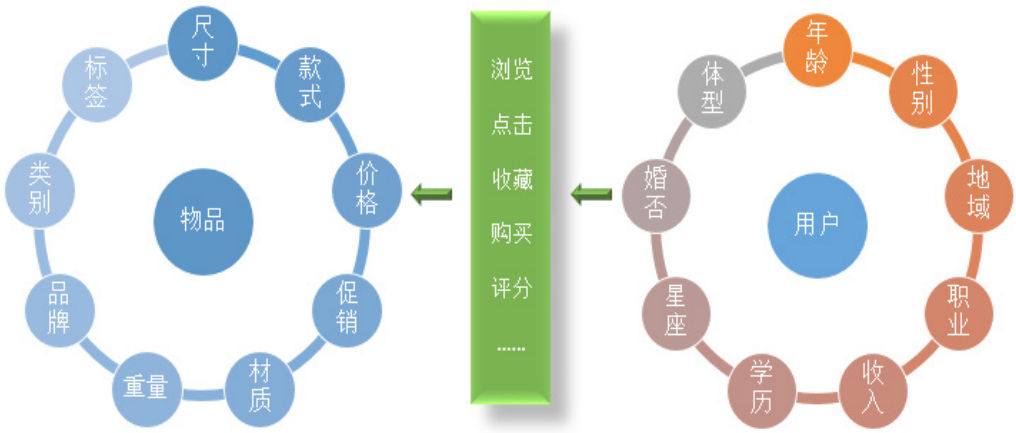

图 2

用户的每一次的行为操作无不反应用户内心的本质需求，包括页面浏览、点击、收藏、购物、搜索、打分、评论等，这些数据是个性化推荐系统的最重要的数据。根据用户自身独有的行为数据，可以为每一个用户生成特有的画像（图 2）。

数据的可靠性很大程度上决定了推荐系统的准确性。原始的数据源中会混杂各种各样的脏数据，一方面是在数据采集和上报的过程中，出现一些异常数据，另一方面也包括系统上线运行时所产生的作弊数据。

数据挖掘的首要步骤就是对原始数据的深度清洗。良好的数据预处理可以对个性化推荐的效果起到事半功倍的效果。数据采集和上报的异常数据，需要结合数据库表结构和实际场景做过滤，如空值检查、数值异常、类型异常、出现“\r”和“\n”导致的字符串换行、数据去重。另外，对于 “人为”的脏数据，如刷点击、刷榜单等行为，这些关键数据会严重影响后续算法的效果，需要有一些反作弊策略进行清除或者降权，如进行 session 分析，结合 cookie、ip、行为发生的时间和次数等一些规则进行过滤。达观数据在反作弊方面也做了很多工作，包括基于机器学习的方法和一些强规则，可有效筛选各种行为上的作弊情况。

2. 用户行为建模

基于用户历史行为的进行挖掘分析，得到刻画用户本质需求的一组属性集合，即得到用户模型，个性化推荐的准确性很大程度上依赖于对用户属性刻画的准确性。达观数据采用了多种方式进行量化，主要包括用户偏好分析和深层用户兴趣点挖掘。（达观数据联合创始人于敬）

1) 用户偏好分析

结合用户历史行为和物品信息，可以得到每种行为下的用户偏好数据，包括偏好的维度及偏好程度，如偏好的物品、品牌、类别、标签等。

$$bias(i,j) = \frac{count(i,j)}{\sum_{k=1}^{n} count(k,j)}$$

再将各种行为的偏好数据合并，最终得到用户在物品、品牌、类别、标签等各个维度上的偏好程度。合并不同维度的数据时，需要考虑到不同的行为类型反应用户偏好程度是不同的。比如购买行为比点击行为更能反映用户的偏好，则由购买行为计算得到的偏好数据在合并时赋予的权重要高一些。

$$bias(i) = \sum_{k=1}^{n} \beta_k * bias(i,k)$$

要保证各种行为的各个维度的数据具有可比性，需要进行归一化，而且同纬度的要采用相同的归一化方法。比如常见的 min-max、log 函数转换、z-score

处理均值问题时，会碰到一个置信度相关度问题。如 10 张投票中有 1 张赞成票和 100 张投票中有 10 张赞成票，虽然均值的角度来说都是 0.1，但是后者的可信度更高。

$$\frac{\hat{p}+\frac{1}{2n}z^2_{1-\frac{\alpha}{2}}-z_{1-\frac{\alpha}{2}}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}+\frac{z^2_{1-\frac{\alpha}{2}}}{4n^2}}}{1+\frac{1}{n}z^2_{1-\frac{\alpha}{2}}}$$

对于样本较多的情况(np > 5且 n(1-p) > 5),可以使用“正态区间”,样本少的话准确性就很差。解决小样本的均值问题,可以基于“威尔逊区间”,使用它的下限值可有效降低“赞成票比例”的作用。

2) 深层用户兴趣点挖掘

除了结合物品信息进行分析计算得到的偏好信息外,还有一部分深层兴趣点需要挖掘,这部分主要用于对细分的用户群体进行更有针对性、更有效的推荐。划分群体的准则要根据具体的业务需求而定,比如是否是高价值用户、是否价格敏感、是否对大牌情有独钟、大神用户和小白用户的区分、喜欢热门流行还是偏小众的,等等。

借助机器学习中的分类(如神经网络、Random Forest、SVM)和聚类(如FCM、k-means)算法可有效解决用户群体的划分问题。使用到的训练和测试数据需要先根据一些规则粗略得到候选集,在结合人工标记的进行筛选。同时从物品和用户的属性数据中也能抽取大量关键特征。经过模型的训练、测试和后处理,从而将用户划分到不同的群体。

3) 计算相似用户

在个性化推荐中,应用很广泛的是基于用户的协同过滤算法。具体又分为 item-based 协同过滤和 user-based 协同过滤,其中 user-based 协同过滤算法核心思路是相似用户的计算。相似度的距离计算公式包含多种方法,如余弦相似度、皮尔逊相关度等。不同的相似度计算方法有各自的优缺点,适用不同的应用场景,可以通过对比测试进行选取。达观数据的实践经验里,一般来说,在基于用户的推荐系统中,皮尔逊相关系数效果要好一些。基于物品的推荐中,余弦相似度方法比皮尔逊的表现更好。

在实际业务中，相似度的计算方法都有很多变种，过于冷门和过于热门的物品对衡量用户间的相似度时区分度不好，这时就需要进行归一。下面的相似度计算方法将降低热门物品对用户相似度的影响。

$$W_{uv} = \frac{\sum_{i \in N(u) \cap N(v)} \frac{1}{\log(1 + |N(i)|)}}{\sqrt{|N(u)||N(v)|}}$$

这种基于 K 近邻的选取相似用户的方法，相似度的阈值设置对结果影响很大，太大的话召回物品过多，准确度会有下降。

4) 时间维度上的考量

在处理各个维度的偏好数据时，需要考虑用户行为的有价值程度是随时间衰减的，即行为发生时间距当前的时间越近，得到的数据越能表征用户将来的行为。毕竟用户的口味随着时间的推移是会变化的，所以时间越近权重越高。衰减函数可以选择离散型或者连续性的。

另外，还需要考虑偏好和兴趣点数据的在时间上的持续和变化过程，即需要刻画用户的口味呈现的时间规律。为了解决这个问题，可以根据不同的时间间隔来界定，分长期、短期、近期和实时四个时间维度。长期的覆盖了用户几乎一直不变的兴趣，短期的覆盖了用户变化中的兴趣，而近期则反映了用户的“尝鲜”的特点。这三种兴趣是离线计算的，此外还要考虑用户的实时兴趣，实践中可以通过很短的时间间隔进行近线挖掘分析，从而快速适应用户当前的信息需求。比如选取最近 1 个月、最近 1 周和最近 2 天来界定长期、短期和近期。

3. 个性化推荐的实践经验

基于准确的用户模型和原始物品属性数据，使用多种推荐算法和效果优化策略，个性化推荐系统将用户最感兴趣的物品列表精准推荐给当前用户。由于不同算法有自己的应用场景，需要根据业务需要、数据的丰富程

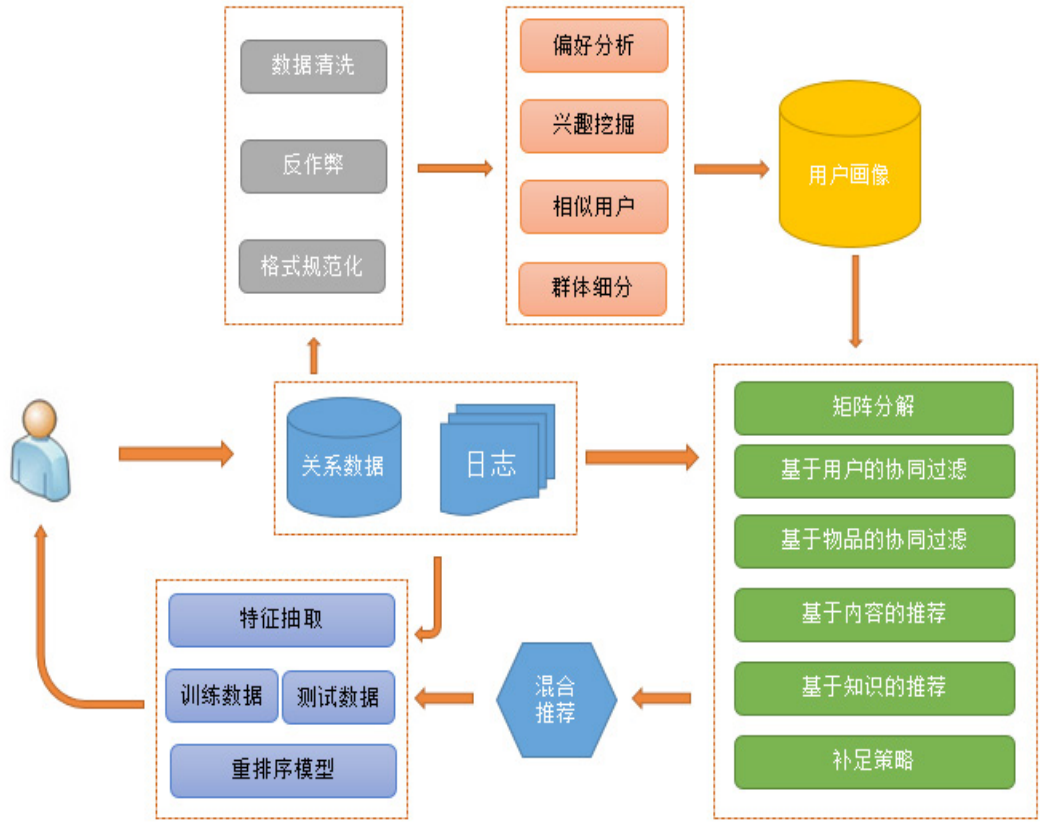

图 3

度、效果衡量指标等选择合适的推荐算法，然后根据推荐结果进行不断迭代，最终完成符合预期效果的个性化推荐系统。（图 3）

1) 基于内容的推荐

主要过程是将用户的信息特征和物品对象的特征相匹配的过程，从而得到待推荐的物品集合。通过用户模型中的类别、标签、品牌等各维度的偏好数据，在全量物品列表中寻找与之匹配的用户感兴趣的物品列表，并给出用户感兴趣的程度。根据挖掘的兴趣点，对部分用户进行有针对性的推荐，为其“量身定制”推荐结果，满足其特有的需求。

基于内容的推荐方法，优点是能保证推荐内容的相关性，并且根据内容特征可以解释推荐结果，而且对新物品的推荐是也能有很好的考量。缺点是由于内容高度匹配，导致推荐结果的惊喜度较差，而且对新用户不能

提供可靠的推荐结果。

另一个重要的优势，是基于内容的推荐能很好的解决推荐系统的“冷启动”问题，因为这类推荐算法不需要依赖用户行为的积累，当待推荐的物品是新出现时，基于内容的推荐算法往往是最有效的方法。

2) 协同过滤

协同过滤方法主要基于群体智慧，认为相似的用户对新物品的喜好也是相似的，相似的物品对于同一用户来说，喜好程度也是相似的。这种方法克服了基于内容方法的一些弊端，最重要的是可以推荐一些内容上差异较大但是又是用户感兴趣的物品。大致分为两类：基于近邻的方法（Neighbour-based）和基于模型的方法（Model-based）。

基于近邻的方法，在数据预测中直接使用已有数据进行预测，将用户的所有数据加载到内存中进行运算。基于用户的协同过滤是，获取和当前用户相似的用户列表，将这些用户喜欢的物品推荐给当前用户。基于物品的协同过滤，获取当前用户偏好的物品列表，将和这些物品相似的物品加入到推荐的候选集当中。

基于模型的方法则是通过数据进行模型训练，然后为用户预测新的物品，主要包括：pLSA(Probabilistic Latent Semantic Analysis)、LDA(Latent Dirichlet Allocation)、SVM(Support Vector Machines)、SVD(Singular Value Decomposition) 等。

推荐系统效果提升的一个有效方式是用户反馈数据的使用，包括显示反馈和隐式反馈。SVD++ 在 SVD 的基础上引入隐式反馈，将用户的点击反馈数据等作为新的参数加入到模型中。

$$r_{ui} = \mu + b_u + b_i + \left(p_u + \sum_{i \in Ufeed(u)} \alpha_i y_i \right)^T q_i$$

具体实现可以参考 SVDFeature、LibFM、GraphLab。

在达观数据的实践中，在有充分用户行为数据的条件下进行实验对比，发现基于协同过滤的推荐算法，效果要大大优于基于内容的推荐算法，但是这也依赖于对用户行为数据的积累和充分清洗。

3) 基于知识的推荐

当用户的行为数据较少同时又有明确的需求时，协同过滤和基于内容的推荐效果不尽人意，但是基于知识的推荐（Knowledge-based Recommendation）可以帮助我们解决这类问题。这种方法不需要用户行为数据就能推荐，依赖人工的先验经验，所以也能较好的处理冷启动问题。

基于知识的推荐其基础是知识库，或称为 Ontology。与其相关联的技术包括自然语言处理、语义分析等。例如“手机”和“手机壳”，“笔记本电脑”和“macbook”，在知识库概念里存在“本体”、“实例”、“关系类型”等抽象概念，利用知识库的延伸或拓展可以形成若干规则、或相互关系，并进而形成推荐体系。推荐结果主要依赖两种形式，一是用户需求跟物品之间相似度，一种是明确的推荐规则。实际应用主要是以强规则为主。在达观科技的实践中，常用于少量特殊场景下的推荐修正。

4) 推荐补足策略

当用户历史数据比较局限或者在冷启动的时候，导致待推荐物品的数量不足没有达到预定要求时，根据用户模型的数据，结合挖掘的各种榜单进行补足，如全局热门、分类热门、潜力物品、潮流物品等。

推荐补足的出发点有很多，第一是确保推荐结果的多样性，因为用户的偏好往往是多样的，因此需要多样的结果来分别满足。单一的推荐算法往往推出的结果较为单调，而通过特定的补足策略可以照顾到多样性的需求。第二是解决推荐的新颖性问题。通常较为小众的物品往往难以得到有效的曝光，或者新出现的内容由于积累的用户行为数据少，往往被压制，

适当的进行这类结果的补足可以较好的提高推荐结果的新颖性。

5) 多算法融合

前文提到单一算法有各自的优缺点，并不能满足实际的线上需求。所以为了提供最优质的个性化推荐服务，保证推荐结果的多样性、新颖性和惊喜度，需要融合多个推荐算法，进行混合推荐。常见的混合方法有以下几种：

a. 加权式混合

主要是对每个算法赋予不同的权重，通过将多个推荐算法的结果进行加权组合在一起，最后排序得到推荐结果。

$$rec_{weighted}(u,i) = \sum_{k=1}^{n} \beta_k * rec_k(u,i)$$

不同推荐算法的结果需要归一化在相同的范围内，并且各个算法的权重之和为 1。

b. 交叉式混合

主要是直接将不同的推荐算法的结果组合在一起推荐给用户，从而每

$$rec_{mixed}(u,i) = \bigcup_{k=1}^{n} \langle rec_k(u,i), k \rangle$$

个推荐算法的优质结果都会被展示给用户。

c. 切换式混合

主要是根据不同应用场景决定使用哪一种推荐算法，应用场景改变的话则切换推荐算法。例如在新闻推荐时，首先使用基于内容的推荐，当找不到合适的内容时，接着使用协同过滤算法进行跨内容的推荐，最后使用朴素贝叶斯分类器找到与用户长期兴趣匹配的结果。

d. 串联混合

主要是将不同的推荐算法进行排序，后面的推荐算法对前面的不断优化，最终得到一个多级优化下的推荐结果。

e. 分级混合

主要是先界定不同的算法的好坏，优先使用好算法的推荐结果，得不到结果时再使用次好的，依次类推。

达观数据在实践中充分利用了各种混合方法来提高推荐效果，并取得了优异的成效。例如基于加权式和分级混合的流程是，首先通过权重的大小来衡量每种推荐算法结果的好坏，产生待推荐的物品集合，在合并的时候，将优先使用好的推荐算法的结果。关于算法权重的设置，基于用户反馈数据，简单的方法可以使用 LR（Logistic Regression）。实践中则是各种指标综合权衡，整个过程也要复杂很多。

6) 重排序

排序学习 (Learning To Rank, LTR) 一直是机器学习中的热门研究领域。由于排序过程牵涉到各种维度的参数数据，导致调参费时费力，而且很可能会出现过拟合现象。而机器学习方法不仅有成熟的理论基础，而且很容易融合多种特征，通过不断的迭代来进行参数优化，可有效解决数据稀疏、过拟合等问题。著名的 Netflix 公司就在他们的推荐系统中全面应用了 LTR 技术（图 4）。

对于已标注的训练集，首先选定 LTR 方法，确定损失函数，以最小化损失函数为目标进行优化即可得到排序模型的相关参数，这就是学习过程。预测过程将待预测结果输入学习得到的排序模型中，即可得到结果的相关得分，利用该得分进行排序即可得到待预测结果的最终顺序。LTR 分按点 (pointwise)、按对 (pairwise) 和按表 (listwise) 三种方法，涉及到的常见模型有 LR(Logistic Regression)、SVM、DT(Decision Tree)。

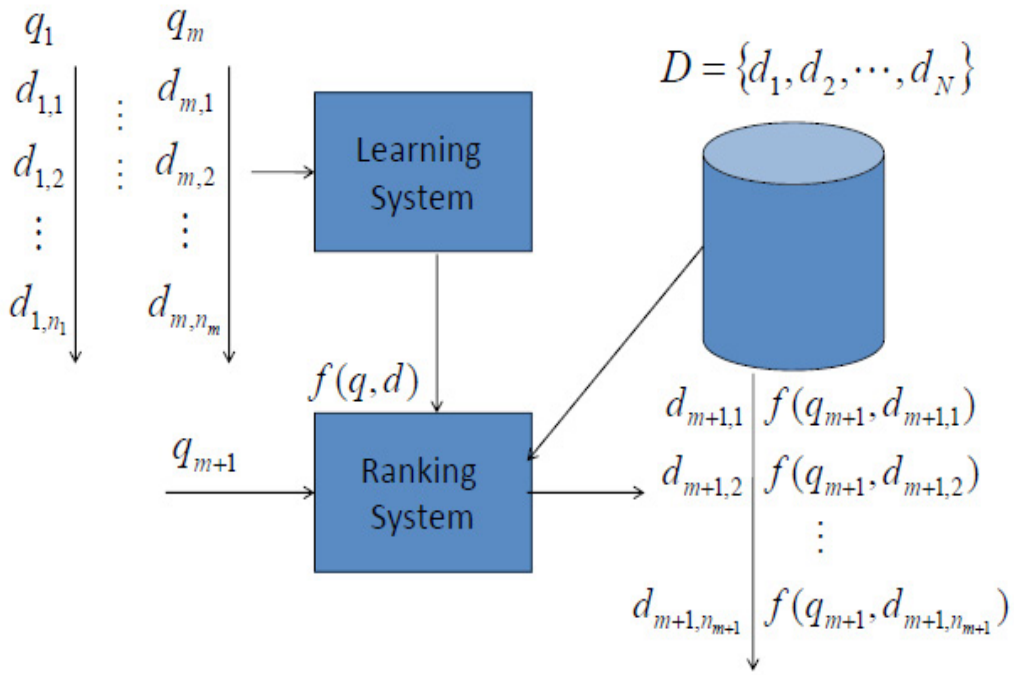

图 4

关于排序模型的选择，LR 算法主要适用于特征数很多、样本量很大的情况。如果是样本量很大，但是特征比较少的情况时，建议使用 DT 的算法。主要是因为在特征数较少时，对应的问题往往是非线性的，此 DT 算法可以发挥自身的优势。另外，SVM 在解决非线性分类问题是效果也非常好。相对于另外两种方法，按表的方法往往更加直接，它专注于自己的目标和任务，直接优化排序结果，因此往往效果也是最好的。

经过多个推荐算法的处理，最终得到待推荐物品的结合，使用少量维度的特征进行排序过于简单，效果也大打折扣。基于推荐算法得到的相关特征，结合物品和用户的特征进行组合，可以得到各种特征，并且有些特征是正相关有些是负相关，需要不断优化。借助机器学习方法得到了最终的物品排序，呈现给用户。

4. 结束语

本文从基本的数据处理、构建用户模型到个性化推荐，介绍了达观数

据的一些实践个性化推荐方面的经验。个性化推荐系统可以有效解决信息过载和长尾物品两个方面的问题，不仅提供了极佳的用户体检，满足了用户的信息需求，也帮助了企业充分利用其中蕴含的商机，提升经营业绩。达观数据一直致力于为企业提供优质的大数据服务，经过多年的积极探索，目前在个性化推荐系统研发和效果提升方面已经积累了丰富的实战经验。当然新技术也在不断出现，深度学习的兴起也给个性化推荐效果的提升带来了更大的契机和想象空间，达观数据也在这方面进行不断探索和测试，后续有机会再跟大家一起分享。不同的细分类型以及其优点和缺点。在第二章和第三章中，我们将会详细介绍这些算法的区别，让你能够深入理解他们的工作原理。系列文章中的一些内容参考了一篇来自 RecSys 2014 tutorial 的文章：由 Xavier Amatriain 编写的 The Recommender Problem Revisited。

作者简介

于敬，同济大学计算机应用技术专业硕士，现任达观数据联合创始人，曾在盛大创新院智能推荐组负责大数据分布式处理、数据挖掘和分析、智能推荐，在盛大文学数据中心负责用户行为建模、个性化推荐、大数据处理、数据挖掘和分析、文本智能审核、反作弊和广告投放引擎。对智能推荐、数据挖掘、大数据技术和广告引擎有较深入的理解和实践经验。

版权声明

InfoQ 中文站出品

出版：极客邦控股（北京）有限公司

北京市朝阳区洛娃大厦 C 座 1607

欢迎共同参与 InfoQ 中文站的内容建设工作，包括原创投稿和翻译，请联系 editors@cn.infoq.com。

网 址：www.infoq.com.cn