在 Quora 做机器学习「炼丹」是种怎样的体验?

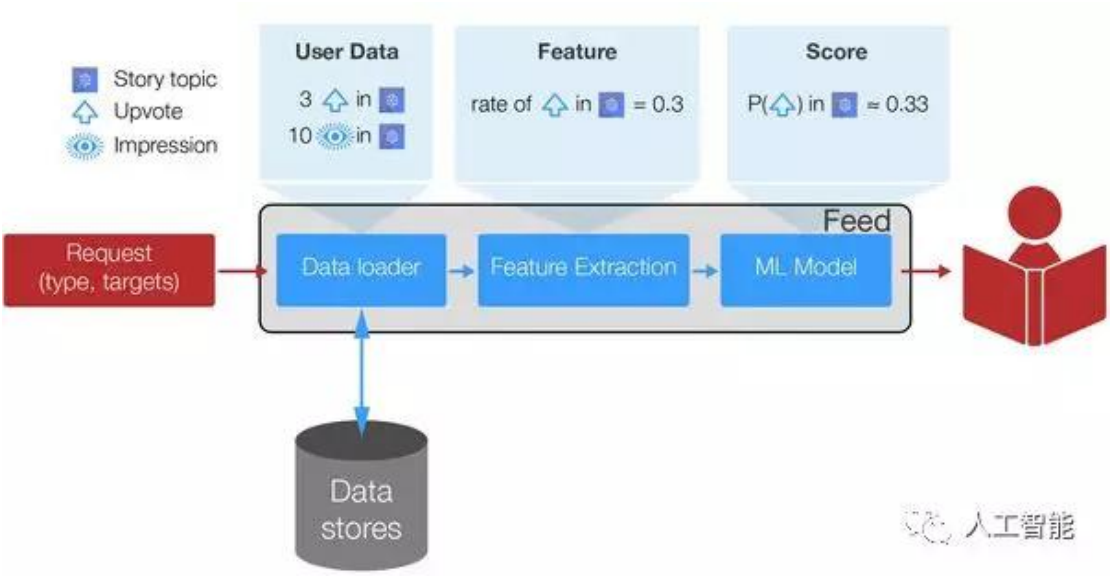
人工智能2019-02-28

实际上,号称「美版知乎」的 Quora 也已经大量引入了机器学习技术,而 Quora 的工程师们则喜欢把自己研究机器学习、产出技术方案的过程戏称为「炼丹」,如今他们也想对外分享他们的经验和成果,开始做一系列「机器学习炼丹之旅」的技术博客。日前,他们发出了该系列博客的第一篇——《使用 Alchemy 做特征工程》(「Feature Engineering at Quora with Alchemy」),作者为 Quora 的两位工程师 Kornél Csernai 和 Naran Bayanbat。雷锋网(公众号:雷锋网) AI 科技评论编译如下。

概述

一直以来,Quora 都致力于拉近人们和有益知识的距离。这意味着,我们需要不断增加知识的采集,并对知识进行评估,还要知道怎样高效地对它们进行排序和分类。现在,Quora 的重点便是通过问答来实现这些目标:读者在网站上寻找相关的内容,提问者寻找有用的答案,而我们就是从他们的提问中提取出正确问题的答复者。其次,我们也需要考虑检测并删掉违规的内容,以及识别出重复的问题。而为了实现这些目标,我们重点倚靠机器学习技术。

首先让我们来看一个典型的案例:Quora 的 Home Feed 功能。在这个最简单的功能中,系统会根据我们认为特定用户可能重视的上下文中的一系列特征,对候选 post 进行排序。之后,这些计算了得分的输入就会形成一个基本的漏斗(funnel)。



一个强大的特征能够明显提升产品的质量，因此快速迭代模型特征非常重要。随着 Quora 网站上机器学习工程师数量日益增多，用来支持产品中各类应用程序的独一无二的特征工程框架的数量也在增多。这种增长自然也带来了一些挑战：

- 一如既往地，更多的框架意味着更多碎片（fragmentation）、更多冗余（redundancy），以及给我们的工程团队带来更重的维护负担。各个团队的开发人员正在创建不同的系统来应对同一个技术挑战，并且无法跨多个应用程序重复利用特征。
-
- 对于一个典型的机器学习应用程序来说，针对每个问题都有成百上千篇候选 post，而每篇候选 post 中又可以提取出数百个特征。同一时间，多个产品服务又必须要实时响应，且响应的时间不能超过几百毫秒。这就意味着，特征提取受限于时间很紧的速度、计算以及记忆局限。我们其中的一些框架难以应对这些局限，也无法成功从日复一日的特征开发中将系统所面临的挑战抽象化。
-

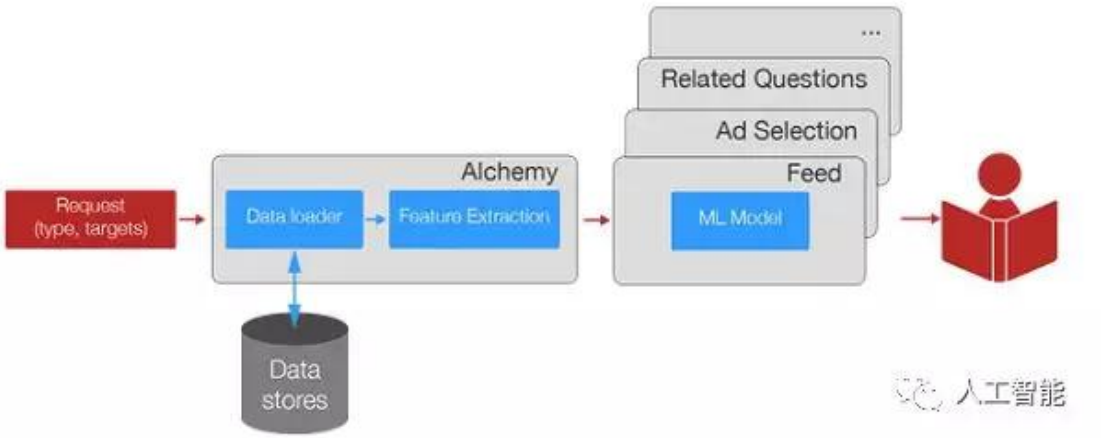
高水平的设计

为了应对上述所有的挑战，我们为特征提取开发了一个高性能、规模化、无国界的服务——Alchemy，它可以泛化到 Quora 上所有的机器学习相关应用程序。

在这 Feed 功能的案例以及相似的应用程序中，这就意味着将特征提取这一步骤单独分离出来作为该网站的一项服务。

之 前 :

之 后 :



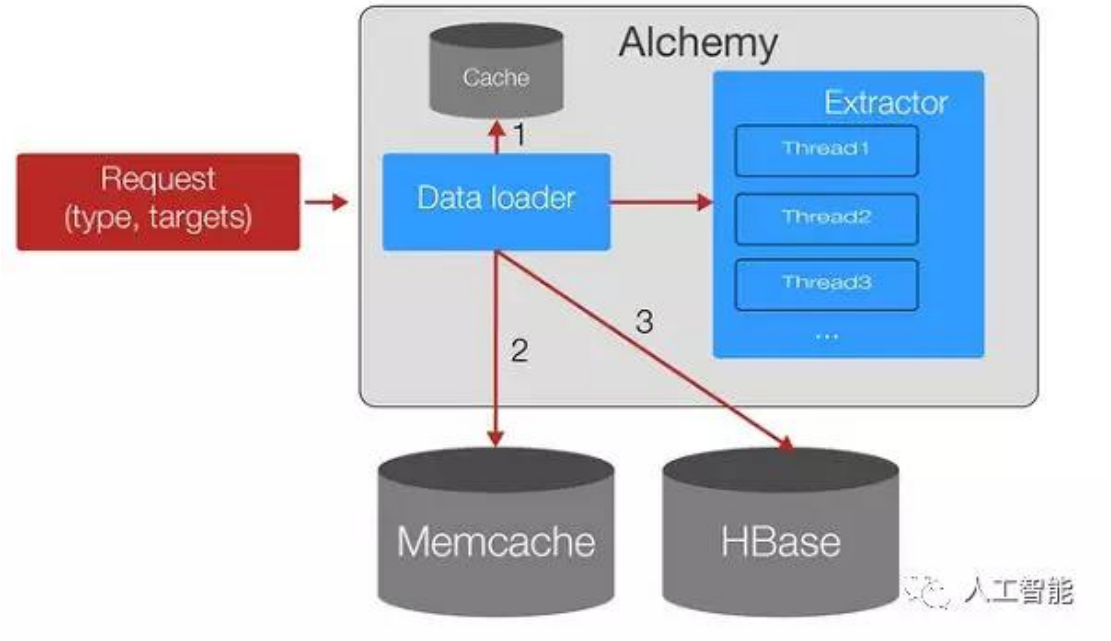
Alchemy 使用应用程序类型（如 Feed 功能）和目标列表（如候选 post），就能为它们各自返回一个数值化的、经过分类的特征向量。假设将特征组织成逻辑组（每个逻辑组表示为一个类别）的形式，这种形式具有数据依赖关系（data dependencies），例如一个问题所对应的答案数量。之后，应用程序就会确定应该提取哪个特征组。

来自应用程序的请求会实时排列，并且在某些情况下，源于不同期工作的请求也会进行排队。每个请求包含了需要评估特征的请求类型以及候选列表（同时，相应的应用程序会定义每个特征列表）。之后，Alchemy 检索并组织任一必要的数据，并最终平行计算出每个目标的特征。

数据存储

Alchemy 使用 HBase 作为它的存储层，不过它也可以轻易地进行扩展来支持其他的存储库。为了让查询更加高效 ,我们采用了多个缓冲层(HBase 顶部的 Memcache 以及 Memcache 顶部的一个内存 LRU cache）。我们的数据由 Thrift 目标组成，它们可以直接存储于内存 cache，以及以二进制形式（binary serialized format）存储于 HBase 和 Memcache。

由于高速缓冲器存储的值会变得陈旧，我们就需要让 Alchemy 知道何时需要从 ground truth 中舍弃掉一些值。这项操作可以通过这种方式实现：从应用程序端写入 Kafka 队列并从 Alchemy 端上的这一队列中读取值。该队列包含 table、key 以及值信息，因此 Alchem 可以在内存缓冲器中用新的值来取代旧的值。



性能

实现 Alchemy 的第一个版本后，我们投入了大量精力来提高该服务的性能。为了让 Alchemy 变得更加快速，我们采用 C++语言将其写成了一个单独的服务，这与将特征提取器嵌入 web 应用程序的代码中的做法相反。这样做的原因在于：

- C++ 是一个低水平的编程语言，它通常可以比用 Python 写的应用程序运行得更快。
-
- 针对每个问题，我们都会发送成百上千篇候选 post 进行评估，代码在候选级别上就能轻松实现并行。而在 Python 中，代码很难在一个过程内实现并行。
-
-

将 Alchemy 写成一个单独的服务，我们就能够在存储器中存储大量反序列化的目标，从而巨大地减少数据检索的延迟（latency），然而这些在 web 应用程序代码中通常是无法实现的。

-

除了决定采用这些设计，我们还 profile 了代码以找到并消除热点。此外，我们还不断审查了大量的度量标准，以了解当这些热点出现时的速度回归（speed regression）源。

（是的，当为单个特征提取编写代码时，选择采用 C++ 可能会降低开发速度。然而，我们认为 C++ 所带来的优势要远大于这一损失，因为机器学习工程师在使用 Alchemy 时不再需要担心性能优化问题。）

迁移和未来工作规划

到目前为止，我们对 Alchemy 所做的所有迁移（migration）都带来了积极的结果。对于我们的在线预测系统，请求延迟得到改善之后，我们就有了更多的空间去尝试那些比较花时间、计算成本比较高的特征。此外，我们可以对更多候选 post 进行排序，从而提供更好的用户体验。使用 Alchemy 还让我们能够继续维护以前耗费过多时间去维护的离线系统（其中的一些离线系统已经被我们在线迁移了）。总的来说，我们已经能够看到，工程师可以更频繁地启动特征并迭代他们的模型。

这些迁移还带来了许多观点、反馈和特征请求，它们都将以实质性的方式积极为我们的路线图提供信息。举一个小例子，我们开始对目标 ID 的请求进行分区以实现更好的缓存利用率。

至于未来的工作规划，我们可能会增加 Python 的绑定，以便工程师和数据科学家可以更广泛地使用该系统。我们还会识别特征之间的依赖关系，构建特征图并异步计算出特征和数据的依赖关系。一个灵活的异步数据检索抽象，会让访问多个数据的存储变得更加容易。所有这些改进都可以将机器学习工程师的负担转移出去，从而使他们可以更专注于开发出色的机器学习模型。

结论

一旦 Alchemy 能够结合大量的应用程序，它的投资回报率是非常高的。系统特征提取做得更好，也意味着它的排序也做得更好。

所有人都说，增长是一个伴随着挑战的好事。弄清楚如何明智地控制这种增长以减少低效率和冗余，是值得从多个维度深入思考的事情，并且这种增长可能会带来很大的收益。