Alibaba Group 阿里巴巴集团 | 阿里技术

2017 阿里技术年度精选（下）

- 阿里前沿技术精华总结
- 覆盖多个热门技术领域
- 10+位技术大牛独家访谈

赠阅

阿里技术

扫一扫二维码图案，关注我吧

「阿里技术」微信公众号

「阿里技术」官方微博

Alibaba Group 阿里巴巴集团 | 阿里技术

2017 阿里技术年度精选（下）

阿里技术

序

你好，我是阿里妹，第一次这么正式地写序，有点紧张。

2017 年，在技术发展的历史上，一定是个特别的一年：柯洁与 AlphaGo 的惊世大战，无人咖啡店开放体验，AI 设计师“鲁班”横空出世、三年投入千亿的达摩院正式成立……

技术前进的脚步，比我们想象中更快。同样在今年，阿里技术公众号，正式迎来了第 35 万位关注者，也有幸发布过数篇 10w+ 乃至百万阅读的文章。

PS：本想借此搞个联谊活动，瞅瞅后台性别比例 9:1，又默默放弃了。

有读者留言告诉我们，每天早上坐公交车上班，看阿里技术推送的文章，已经成为了一种习惯。这让我们倍感不胜荣幸，又更觉肩上责任之重。

我们始终相信，每天一篇干货，效果虽不能立竿见影，但聚沙成塔、滴水成川，你看过的东西，会成为你的一部分，潜移默化影响更多的人。

阿里技术公众号，看似只有阿里妹一个人运营。但在它身后，其实有超过 25000 名阿里工程师的力量。这群可爱的作者里，有刚走出校门的新人，有团队中的中流砥柱资深工程师，有带领数百、上千人的技术管理者。他们在工作之余，抽出宝贵的休息时间，梳理自己对业务、技术、人生的思考和理解。

没有华丽的词藻、繁复的修辞，只有朴质的代码、反复推敲的算法。一篇文章看似只有几千字，但在背后，往往是数十人乃至数百人的团队，长达数月、数年的摸索、跌倒，终于探得的成果。他们把这一路的所得，迫不及待地分享给业界同仁，希望帮助大家少踩坑，少加班。

这次，在全年发布的近 300 篇文章中，我们选出 65 篇，集结成这套《2017 阿里技术年度精选》，分为上、下两册，总计 600 余页。

这套精选集覆盖多个热门技术领域：算法、机器学习、大数据、数据库、中间件、运维、安全、移动开发等，文章内容涉及技术架构、核心算法、解决方案等干货。无论你是计算机相关专业的在校学生、科研机构的研究人员，还是步入社会的 IT 从业人员，相信都能从中受益。

这套书同时收录了十多位阿里技术人的访谈：从工程师到合伙人的多隆，6 年时间影响数亿用户的靖世，入选 MIT2017 年度 TR35 的王刚 & 吴翰清，免试晋升为研究员的钱磊等，将为你展现不一样的技术人生。

它不是一本系统讲述某个领域的书，更像是一本技术杂文选集，内容五花八门。翻开书来，一眼望去，皆是散落在各个技术领域的结晶。你可以把它当作床头书，或是在旅行的路上随手翻翻，充充电。希望这本书，能为你打开一扇窗，去看更大的世界；成为一个小支点，帮你撬动更大的进步。

因为相信，所以看见。这世界依然浮躁，但庆幸始终有人相信，技术让生活更美好。

感谢坚持分享、笔耕不辍的阿里工程师，

感谢所有关注阿里技术的小伙伴，

感谢所有的相遇与陪伴。

希望这套书，能陪你一起回味即将逝去的 2017。

2018，我们一起遇见更好的自己。

最后，阿里妹也期待你的回信，聊聊你的感想与建议：lunalin.lpp@alibaba-inc.com

阿里妹

2017 年 12 月

目录

AI/ 算法

60 年后的你长什么样？人脸老化三大技术探秘

塞远

通过计算机视觉技术模拟出人老了的样子，这样的应用实际上并不新奇。这些年，类似的场景可谓层出不穷，有的 App 可以通过夫妻的样貌预测出孩子的模样，有的 App 可以通过用户上传的照片还原你小时候的样子和老了之后的样子，甚至还有一些 App 可以通过一两张照片生成出一段短视频，展示用户一生的样貌变化，让使用的人颇有沧海桑田之感。

而我们这次的“当你老了”项目的着眼点略有不同，强调的是公益和技术的结合，因此我们需要将用户上传的面部照片变成 20 年，40 年和 60 年后，让用户看到自己满脸皱纹，白发苍苍的样子，进而产生关爱身边老年人的感慨，人人每年公益 3 小时，共同为公益做出贡献。

经过将近 4 周左右的努力，我们初步完成了人脸老化项目的开发，赶在 9 月 5 日央视慈善之夜晚会前及时上线。央视公益晚会期间，用户通过扫描电视屏幕上的二维码，进入我们手淘的互动页面（详见图 1），选择自己喜欢的照片进行老化。一场晚会下来，许多的用户体验到了自己脸部变老之后的特效。

图 1　手淘“当你老了”项目页面示意图

整个项目中我们遇到了很多的挑战，比如老化模版的选择，脸型的变化趋势问题，肤色的融合问题等等，下面我会把整个人脸老化技术分成几个关键的部分，进行简短的剖析。

1. 脸型的老化算法

我们借鉴了一篇 2017 年最新的利用深度学习进行人脸老化的文章 [1]，该文章提出了一种叫做 conditional adversarial autoencoder (CAAE) 的深度网络结构，该结构能够学习出面部的 manifold，从而预测出任何一张输入面部图像的全年龄面部图像。该网络的结构示意图如图 2 所示，该网络学习之后的预测结果示意图如图 3 所示。

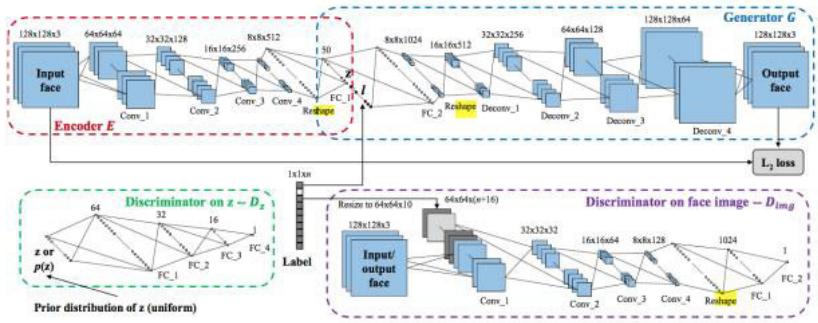


图 2 Structure of the proposed CAAE network for age progression/regression. The encoder E maps the input face to a vector z (personality). Concatenating the label l (age) to z, the new latent vector [z, l] is fed to the generator G. Both the encoder and the generator are updated based on the L2 loss between the input and output faces. The discriminator Dz imposes the uniform distribution on z, and the discriminator Dimg forces the output face to be photo-realistic and plausible for a given age label.

图 3 Comparison to prior works of face aging. The first column shows input faces, and second column are the best aged faces cited from prior works. The rest columns are our results from both age progression and regression. The red boxes indicate the comparable results to the prior works.

尽管该深度学习网络能够很好的获得全年龄段的输出，但是它存在两个弊端，其一是它训练预测出来的图片和输入人脸差异比较大，不像原始输入；其二是处理速度上该网络速度比较慢，很难适应项目的要求。针对这两点缺点，我们决定只保留该网络的老化脸型的能力，而脸上的纹理，即皱纹部分我们通过传统方法来贴。图 4 显示了我们通过该网络获得的脸型变老的能力示意图。

图 4　面部脸型变老的示意图，从左到右分别是原图，20 年后，40 年后，60 年后

用手淘扫描上方二维码，遇见 60 年后的你

2. 脸部纹理的老化算法

在贴脸部皱纹或者说纹理的过程中，我们采用的是分而治之的办法 [2]。即把面部分成许许多多的小三角形，然后将老人模版上对应的三角形贴到相应的输入人脸的三角形上。这样做的好处是每个三角形足够小，利用 OpenCV 的 WarpAffine 进行形变贴图时不会出现不自然的现象。当然这样做的前提是，我们已经获得了输入人脸以及模版人脸的所有关键点，同时利用 Delaunay Triangulation 将所有关键点划分成若

干三角形了。关键点和三角形 Mesh 示意图如图 5 所示。

图 5　面部关键点示意图，面部三角形化示意图

3. 肤色的融合算法

在完成了人脸的老化脸型变化和贴皱纹的步骤后，我们还缺最后一步，即将老化后的脸完美无瑕地贴回到原始输入图像上去，否则会显得老化很不自然。这里我们采用了一个非常经典地融合算法，即 Poisson Image Editing[3]，它是通过保留剃度的方式使贴过来的图像能够无缝地融入背景图像。这里我们利用了 OpenCV3.2 里面提供的 SeamlessClone 这样一个函数，可以比较方便地实现前后景融合。肤色和纹理的融合示意图如图 6 所示。

图 6　从左到右依次为原始图像，老人模版贴入相应三角形，SeamlessClone 之后的融合结果

4. 项目小结

在脸型，皱纹，肤色的老化之外，我们还尝试了头发的变白等算法，篇幅有限，就不在此赘述了。最后贴几张我们认为效果还是不错的人脸老化结果作为结尾。

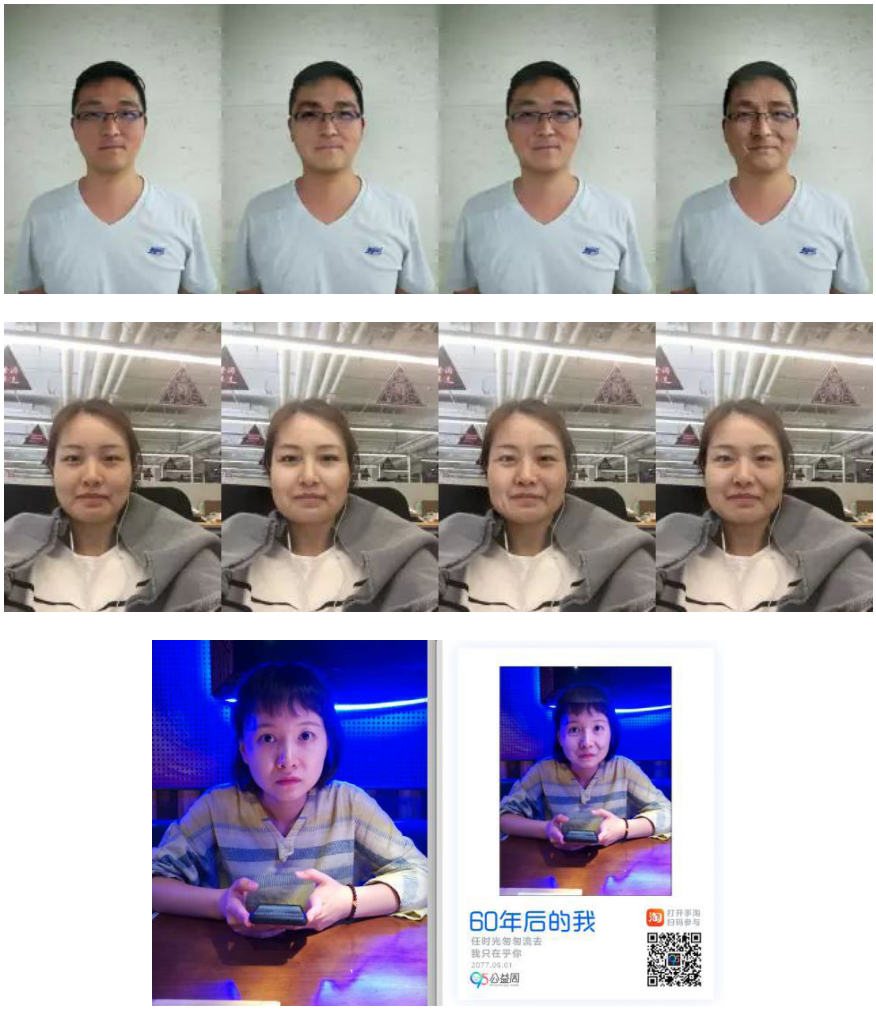


图 7　效果示意图

Reference

[1] Zhifei Zhang, Yang Song, and Hairong Qi. "Age Progression/Regression by Conditional Adversarial Autoencoder." *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[2] Face Morph Using OpenCV — C++ / Python, http://www.learnopencv.com/face-morph-using-opencv-cpp-python/

[3] Poisson Image Editing, http://www.ctralie.com/Teaching/PoissonImageEditing/.

世界级难题：把不同物品装进箱子，如何使箱子表面积最小？

菜鸟人工智能部

阿里妹导读：三维装箱问题是一类经典的组合优化问题，具有巨大的学习研究和实际应用价值。传统的三维装箱问题都是给定了箱子的尺寸并以最小化箱子的使用数量为优化目标，但是在某些实际业务场景中并没有固定尺寸的箱子。

基于此类场景，本文提出了一类新型的三维装箱问题。在本问题中，需要将若干个长方体物体逐个放入一个箱子中（物品的摆放位置不能倾斜），优化目标为最小化能够容纳所有物品的箱子的表面积，因为箱子的表面积与其成本直接正相关。本文证明了此类新问题为 NP-hard 问题。对于装箱问题，箱子的表面积取决于物品的放入顺序、摆放的空间位置和摆放朝向。在这些因素中，物品的放入顺序有着非常重要的影响。所以本文基于近些年被提出的、能够有效解决某些组合优化问题的深度强化学习方法—Pointer Network 方法来优化物品的放入顺序。

本文基于大量实际业务数据对网络模型进行了训练和检验。结果表明，相对于已有的启发式算法，深度强化学习方法能够获得大约 5% 的效果提升。

论文地址：https://arxiv.org/abs/1708.05930

1. 引言

装箱问题是一类非常经典且重要的优化问题，常被应用于物流系统和生产系统中。装箱问题有很多变型问题，其中最重要且最难求解的是三维装箱问题，在此问题中，需要将若干个不同大小的长方体物品放入箱子中，物品之间不能重叠且不能倾斜，箱子的尺寸和成本已知，优化目标为最小化箱子的使用数量，即最小化总成本。装箱问题一直是学术界非常流行的研究方向之一。除此之外，装箱问题在实际中也有很多应用。有效的装箱算法往往意味着计算时间、装箱成本的大量节省和资源使用效率的大幅提升。

在某些实际业务场景中，我们发现装箱时并不是使用固定尺寸的箱子（例如在跨境电商业务中，大部分是使用柔性的塑料材料，而不是箱子来包装货物），而且由于装箱的成本大部分都由装箱材料成本构成，而装箱材料成本又主要取决于材料的表面积。所以在本研究中，我们提出了一类新型的装箱问题。与传统三维装箱问题不同的是，本问题的优化目标为确定一个能够容纳所有物品的箱子，并最小化此箱子的表面积。

由于寻找装箱问题的最优解非常难，所以相关研究者们提出了不同的近似算法和启发式算法来求解此类问题。但是启发式算法往往有着较强的问题依赖性，一类启发式算法可能只适用于求解某种或某些业务场景中的装箱问题。近些年来，人工智能技术，尤其是深度强化学习 (Deep reinforcement learning, DRL) 技术有着非常快速的发展，并且在某些问题上取得了令人瞩目的成果。而且深度强化学习技术已经被发现在求解组合优化问题方面具有较大的潜力，所以本研究使用了一种基于深度强化学习的方法来求解新型三维装箱问题。本文基于大量实际业务数据训练了深度强化学习模型，并验证了模型的效果。

2. 相关研究介绍

2.1 三维装箱问题相关研究

装箱问题是一类非常经典和流行的优化问题。自从其在 20 世纪 70 年代被提出

以来，大量研究者对此类问题进行了研究并获得了许多有价值的成果。[Coffman *et al*., 1980] 证明了二维装箱问题是 NP-hard 问题，所以作为二维装箱问题的一般化问题，三维装箱问题也是 NP-hard 问题。由于此原因，很多之前的研究都集中于近似算法和启发式算法。[Scheithauer, 1991] 首先提出了针对三维装箱问题的近似算法并分析了其近似比。

此外，研究者们还提出了很多不同类型的启发式算法，例如禁忌搜索 ([Lodi *et al*.,2002], [Crainic *et al*., 2009]), 引导式局部搜索 ([Faroe *et al*., 2003]), 基于极点的启发式算法 ([Crainic *et al*., 2008])，混合遗传算法 ([Kang *et al*.,2012]) 等。在精确解算法方面，[Chen *et al*., 1995] 考虑了一种有多种尺寸箱子的三维装箱问题，并建立一个混合整数规划模型来求解最优解。[Martello *et al*.,2000] 提出了一种分支定界算法来求解三维装箱问题，并通过数值实验表明 90 个物品以内的问题都可以在合理的时间内获得最优解。

另外，还有一些从实际业务中提出的装箱问题的变型问题，例如 [Kang and Park, 2003] 提出了一种可变尺寸的装箱问题，[Khanafer *et al*.,2010], [Gendreau *et al*., 2004] 研究了一种考虑物品冲突的装箱问题，[Clautiaux *et al*.,2014] 对一种考虑易碎物品的装箱问题进行了研究。

另一类装箱问题—条带装箱问题 (Strippacking problem) 与本文提出的新问题

比较接近。在一般的条带装箱问题中，若干个长方体物品需要被逐个放入一个给定的条带中，条带的长度和宽度是已知且固定的，长度为无穷大（在二维条带装箱问题中，条带的宽度固定，但是长度为无穷大），优化目标为最小化使用的条带的高度。此类问题在钢铁工业和纺织工业中有很多应用，研究者们也提出了不同类型的求解算法，例如精确解算法 ([Martello *et al*., 2003]，[Kenmochi *et al*., 2009])，近似解算法 ([Steinberg, 1997])，启发式算法 ([Bortfeldt and Mack, 2007], [Bortfeldt, 2006], [Hopper andTurton, 2001]) 等。

2.2 DRL 方法在组合优化问题中的应用研究

虽然机器学习和组合优化问题已经分别被研究了数十年，但是关于机器学习方法在求解组合优化问题方面的研究却比较少。其中的一个研究方向是使用强化学习的思想来设计超启发式算法。[Burke *et al*., 2013] 在一篇关于超启发式算法的综述论文中对于一些基于学习机制的超启发式算法进行了讨论。[Nareyek, 2003] 使用了一种基于非平稳强化学习的方法来更新启发式算法被选择的概率。除此之外，强化学习思想在超启发式算法中的应用研究还包括二元指数补偿 ([Remde *et al*.,2009])、禁忌搜索 ([Burke *et al*.,2003]) 和选择函数等 ([Cowling *et al*., 2000])。

近些年来，序列到序列模型 (sequence-to-sequence) 的一系列研究突破激发了研究者们对于神经组合优化 (neural combinatorial optimization) 方向的兴趣。其中注意机制 (attention mechanism) 对于加强神经网络模型在机器翻译 ([Bahdanau *et al*.,2014]) 和算法学习 ([Graves *et al*.,2014]) 方面的效果中扮演了重要决策。 [Vinyals *et al*., 2015] 提出了一种带有特殊注意机制的网络模型—Pointer Net，并使用有监督学习的方法来通过此模型求解旅行商问题 (Travelling Salesman Problem)。[Bello *et al*., 2016] 提出了一种基于强化学习思想的神经组合优化 (neural combinatorial optimization) 框架，并使用此种框架求解了旅行商问题和背包问题 (Knapsack Problem)。因为此种框架的有效性和普适性，本研究在求解新型装箱问题中主要使用了此种框架和方法。

3. 针对三维装箱问题的 DRL 方法

3.1 问题定义

在经典的三维装箱问题中，需要将若干个物品放入固定尺寸的箱子中，并最小化箱子的使用数量。与经典问题不同的是，本文提出的新型装箱问题的目标在于设计能够容纳一个订单中所有物品的箱子，并使箱子的表面积最小。在一些实际业务场景中，例如跨境电商中，包装物品时使用的是柔性的塑料材料，而且由于包装材料的成本与其表面积直接正相关，所以最小化箱子的表面积即意味着最小化包装成本。

本文提出的新型装箱问题的数学表达形式如下所示。给定一系列物品的集合，每个物品 i 都有各自的长 (l_i)、宽 (w_i) 和高 (h_i)。优化目标为寻找一个表面积最小且能够容纳所有物品的箱子。我们规定 (x_i, y_i, z_i) 表示每一个物品的左下后 (left-bottom-back) 角的坐标，而且 (0, 0, 0) 表示箱子的左下后角的坐标。决策变量的符号及其含义如表 1 所示。基于以上问题描述和符号定义，新问题的数学表达形式为：

$$\min \ L \cdot W + L \cdot H + W \cdot H$$

$$\text{s.t.} \begin{cases} s_{ij} + u_{ij} + b_{ij} = 1 & (1) \\ \delta_{i1} + \delta_{i2} + \delta_{i3} + \delta_{i4} + \delta_{i5} + \delta_{i6} = 1 & (2) \\ x_i - x_j + L \cdot s_{ij} \le L - \hat{l}_i & (3) \\ y_i - y_j + W \cdot u_{ij} \le W - \hat{w}_i & (4) \\ z_i - z_j + H \cdot b_{ij} \le H - \hat{h}_i & (5) \\ 0 \le x_i \le L - \hat{l}_i & (6) \\ 0 \le y_i \le W - \hat{w}_i & (7) \\ 0 \le z_i \le H - \hat{h}_i & (8) \\ \hat{l}_i = \delta_{i1} l_i + \delta_{i2} l_i + \delta_{i3} w_i + \delta_{i4} w_i + \delta_{i5} h_i + \delta_{i6} h_i & (9) \\ \hat{w}_i = \delta_{i1} w_i + \delta_{i2} h_i + \delta_{i3} l_i + \delta_{i4} h_i + \delta_{i5} l_i + \delta_{i6} w_i & (10) \\ \hat{h}_i = \delta_{i1} h_i + \delta_{i2} w_i + \delta_{i3} h_i + \delta_{i4} l_i + \delta_{i5} w_i + \delta_{i6} l_i & (11) \\ s_{ij}, u_{ij}, b_{ij} \in \{0, 1\} & (12) \\ \delta_{i1}, \delta_{i2}, \delta_{i3}, \delta_{i4}, \delta_{i5}, \delta_{i6} \in \{0, 1\} & (13) \end{cases}$$

其中，$s_{ij} = 1$ 表示第 i 个物品在第 j 个物品的左边，$u_{ij} = 1$ 表示第 i 个物品在第 j 个物品的下边，$b_{ij} = 1$ 表示第 i 个物品在第 j 个物品的后边。$\delta_{i1} = 1$ 表示物品 i 的摆放朝向为正面朝上，$\delta_{i2} = 1$ 表示物品 i 正面朝下，$\delta_{i3} = 1$ 表示物品 i 侧面朝上，$\delta_{i4} = 1$ 表示物

品 i 侧面朝下，$\delta_{i5}=1$表示物品底面朝上，$\delta_{i6}=1$表示物品底面朝下。

约束条件中的 (9), (10), (11) 表示物品在不同的朝向情况下占用的空间的长宽高；约束条件 (1), (3), (4), (5) 表示物品之间没有重叠，约束条件 (6), (7), (8) 保证了箱子容纳了所有物品。

基于以上的数学模型，我们使用了优化引擎，例如 IBM Cplex 等来直接求解此问题。但是对于一般规模的问题（例如物品数量大于等于 6），很难在合理的时间内获得最优解。而且我们还证明了此类问题是 NP-hard 问题。证明过程请见附录。

表1　决策变量符号及含义

变量名	类型	含义
L	连续型变量	箱子的长度
W	连续型变量	箱子的宽度
H	连续型变量	箱子的高度
x_i	连续型变量	物品的左下后角的 *x* 坐标
y_i	连续型变量	物品的左下后角的 *y* 坐标
z_i	连续型变量	物品的左下后角的 *z* 坐标
s_{ij}	0-1 二元变量	第 *i* 个物品是否在第 *j* 个物品的左边
u_{ij}	0-1 二元变量	第 *i* 个物品是否在第 *j* 个物品的下边
b_{ij}	0-1 二元变量	第 *i* 个物品是否在第 *j* 个物品的后边
δ_{i1}	0-1 二元变量	第 *i* 个物品是否是正面朝上
δ_{i2}	0-1 二元变量	第 *i* 个物品是否是正面朝下
δ_{i3}	0-1 二元变量	第 *i* 个物品是否是侧面朝上
δ_{i4}	0-1 二元变量	第 *i* 个物品是否是侧面朝下
δ_{i5}	0-1 二元变量	第 *i* 个物品是否是底面朝上
δ_{i6}	0-1 二元变量	第 *i* 个物品是否是底面朝下

3.2　DRL 方法

在本部分中，我们将介绍用于求解新型三维装箱问题的 DRL 方法。在求解三维装箱问题时，决策变量主要分为三类：

（1）物品放入箱子的顺序；

（2）物品在箱子中的摆放位置；

（3）物品在箱子的摆放朝向。

我们首先设计了一种启发式算法来求解此类新型三维装箱问题。此种算法的基本思想为：在放入一个物品时，遍历所有可用的空余空间和物品朝向，并选择能够最小化表面积的组合。然后再遍历所有物品，确定一个能够最小化浪费空间体积（least waste space）的物品。算法的详细步骤请见附录。在本文中，我们使用 DRL 方法来优化物品的放入顺序，在确定了物品的放入顺序之后，选择物品的摆放位置和朝向时使用和以上启发式算法相同的方法。所以本研究的重点在于使用 DRL 方法来优化物品的放入顺序。在未来的研究中，我们将会把物品的放入顺序、摆放位置和朝向统一纳入深度强化学习方法框架中。

3.2.1 网络结构

本研究主要使用了 [Vinyals *et al*.,2015] 和 [Bello *et al*.,2016] 提出的神经网络结构。在 Vinyals 和 Bello 等人的研究中提出了一种名为 Pointer Net (Ptr-Net) 的神经网络来求解组合优化问题。例如，在求解旅行商问题时，二维平面中每个点的坐标被输入到网络模型中，经过计算之后，模型的输出为每个点被访问的顺序。这种网络结构与 [Sutskever *et al*.,2014] 提出的序列到序列模型非常相似，但是有两点不同：第一，在序列到序列模型中，每一步的预测目标的种类是固定的，但是在 Ptr-Net 中是可变的；第二，在序列到序列模型中，在解码阶段通过注意机制将编码阶段的隐层单元组合成为一个上下文向量信息，而在 Ptr-Net 中，通过注意机制来选择（指向）输入序列中的一个来作为输出。

本研究中使用的神经网络模型的结构如图 1 所示。网络的输入为需要装箱的物品的长宽高数据，输出为物品装箱的顺序。网络中包含了两个 RNN 模型：编码网络和解码网络。在编码网络的每一步中，首先对物品的长宽高数据进行嵌入表达（embedded），然后再输入到 LSTM 单元中，并获得对应的输出。在编码阶段的最后一步，将 LSTM 单元的状态和输出传递到解码网络。在解码网络的每一步中，在编码网络的输出中选择一个作为下一步的输入。如图 1 所示，在解码网络中的第 3 步的输出为 4，则选择（指向）编码网络的第 4 步的输出，将其作为解码网络下一步（第 4 步）的输入。此外，在每一步的预测过程中还使用了 [Bello *et al*., 2016] 提出的

Glimpse 机制来整合编码阶段和解码阶段的输出信息。

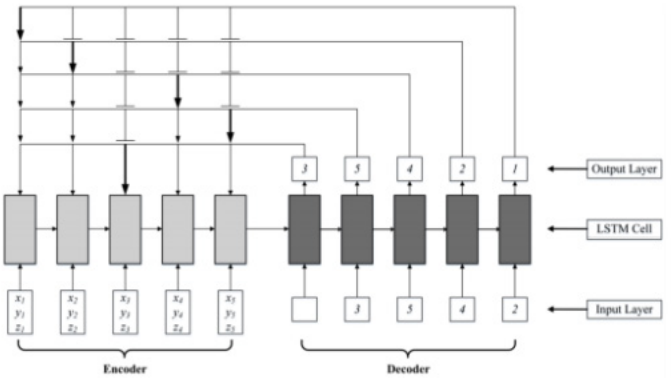


图 1　神经网络模型的结构

3.2.2　基于策略的强化学习方法

本研究中使用了强化学习方法来训练网络模型。网络模型的输入可以表示为 $s=\{(l_i,w_i,h_i)\}_{i=1}^{n}$，其中 l_i, w_i, h_i 分别表示第 i 个物品的长宽高。网络模型的输出为物品放入箱子的顺序，用 o 来表示。我们使用表面积（Surface area，SA）来评价模型的输出结果，使用 $SA(o|s)$ 表示在模型输入为 o，输出为 s 的情况下对应的表面积。模型的随机策略可以表示为 $p(o|s)$，即在模型输入为 s 的情况下，输出为 o 的概率。模型训练的目标为尽可能的使对应表面积较小的输出 (o) 以较大的概率被选中。我们使用 $\boldsymbol{\theta}$ 表示网络模型的参数，则训练目标可以表示为：

$$J(\boldsymbol{\theta}\,|\,s)=E_{o\sim p_{\boldsymbol{\theta}}(\cdot|s)}SA(o\,|\,s)$$

[Williams, 1992] 提出了一种具有普适性的强化学习方法，此种方法能够在训练过程中使模型参数在期望的强化方向上不断地调整。基于此种方法，本研究在训练的每一步中，在获得了奖励值（reward）、基准值（baseline value）和预测的概率分布之后，模型参数的更新公式为：

$$\nabla_{\boldsymbol{\theta}}J(\boldsymbol{\theta}\,|\,\boldsymbol{s})=E_{o\sim p_{\boldsymbol{\theta}}(\cdot|s)}[(SA(o\,|\,s)-b(s))\nabla_{\boldsymbol{\theta}}log\ p_{\theta}(o\,|\,s)]$$

其中 $b(s)$ 表示表面积的基准值，可以用来有效降低训练过程中梯度的方差。在训练过程中，如果我们随机选取 M 个独立同分布的样本 $s_1,s_2,\ldots,s_M$，则以上更新公式

可以近似为：

$$\nabla_{\theta} J(\boldsymbol{\theta} \mid \boldsymbol{s}) \approx \frac{1}{M} \sum_{i=1}^{M}[(SA(o_i \mid s_i) - b(s_i)) \nabla_{\theta} log\ p_{\theta}(o_i \mid s_i)]$$

3.2.3 基准值的更新

在本研究中，我们使用了一种基于记忆重放的方法来不断地更新基准值。首先，对于每一个样本点 *si*，通过启发式算法获取一个装箱方案，并计算其表面积，作为 *b*(*si*) 的初始值。之后在每一步的训练过程中，通过以下公式来更新基准值：

$$b'(s_i) = b(s_i) + \alpha(SA(o_i \mid s_i) - b(s_i))$$

其中 $SA(o_i \mid s_i)$ 为训练过程中获得表面积的值。

3.2.4 随机采样与集束搜索（Beam Search）

在模型的训练阶段，我们从模型预测的概率分布中进行随机选取作为输出。但是在验证阶段，我们采用贪婪策略来进行选择，即在每一步中，我们选取概率分布中概率最大的备选项作为输出。除此之外，我们还在验证阶段使用来集束搜索的方法来提高模型的效果，即在每一步中不是选择对应概率最高的备选项，而是选择概率最高的前 *k* 个备选项作为输出。

通过以上描述，模型的整个训练步骤总结为：

算法1：网络模型的训练步骤	
1:	使用 *S* 表示训练样本集合，*T* 表示训练步数，*B* 表示训练过程中每批样本的样本量
2:	初始化 Pointer Net 的参数 $\boldsymbol{\theta}$
3:	for t = 1 to *T* do
4:	选择一批训练数据 s_i，其中 $i \in \{1,2,\cdots,B\}$
5:	对于 s_i，基于 $p_{\boldsymbol{\theta}}(\cdot \mid s_i)$ 选择模型的输出 oi
6:	计算 $g_{\theta} = \frac{1}{B} \sum_{i=1}^{B}[(SA(o_i \mid s_i) - b(s_i)) \nabla_{\theta} log p_{\theta}(o_i \mid s_i)]$
7:	更新 $\boldsymbol{\theta} \leftarrow ADAM(\theta, g_{\theta})$
8	对于 $i \in \{1,2,\cdots,B\}$，更新基准值 $b'(s_i) = b(s_i) + \alpha(SA(o_i \mid s_i) - b(s_i))$
9:	end for
10:	返回模型参数 $\boldsymbol{\theta}$

4. 数值实验

为了验证模型的效果，我们基于大量实际业务数据完成了数值实验。根据实验过程中每个订单中物品数量的不同（8，10 和 12），实验分为了三个部分，但是每次实验过程中的超参数均相同。在每次实验中，我们采用了 15 万条训练样本和 15 万条测试样本。在实验过程中，每批训练的样本量为 128，LSTM 的隐层单元的数量为 128，Adam 的初始学习速率为 0.001，并且每 5000 步以 0.96 的比例衰减。网络模型的参数的初始值均从 [−0.08, 0.08] 中随机选取。为了防止梯度爆炸现象的出现，我们在训练过程中使用 L2 正则方法对梯度进行修剪。在更新基准值的过程中，的取值为 0.7。在每次训练中，我们在 Tesla M40 GPU 上训练 100 万步，每次的训练时间大约为 12 小时。在验证阶段，使用集束搜索方法时，集束搜索的宽度为 3。模型主要通过 TensorFlow 来实现。

数值实验的结果请见表 2。主要评价指标为平均表面积（Average surface area, ASA）. 从表中可以看出，在使用集束搜索的情况下，本文提出的基于 DRL 的方法在三类测试集上分别可以获得大约 4.89%, 4.88%, 5.33% 的效果提升。此外，我们还通过穷举的方法获得了对于 8 个物品测试数据中 5000 个样本数据的最优物品放入顺序，并计算得到了启发式算法的结果与最优解的平均差距为 10% 左右，这说明基于 DRL 的方法的结果已经与最优解比较接近。

表2　不同方法下获得的ASA

物品数量	随机方法	启发式算法	深度强化学习方法（随机选取）	深度强化学习方法（集束搜索）
8	44.70	43.97	41.82	41.82
10	48.38	47.33	45.03	45.02
12	50.78	49.34	46.71	46.71

5. 结论

本文提出了一类新型三维装箱问题。不同于传统的三维装箱问题，本文提出的问题的优化目标为最小化能够容纳所有物品的箱子的表面积。由于问题的复杂性和求解难度，此类问题非常难以获得最优解，而大部分启发式算法又缺乏普适性。所以本

文尝试将 Pointer Net 框架和基于深度强化学习的方法应用到了对此类问题的优化求解中。本文基于大量实际数据对网络模型进行了训练和验证，数值实验的结果表明基于深度强化学习方法获得的结果显著好于已有的启发式算法。本项研究的主要贡献包括：第一，提出了一类新型的三维装箱问题；第二，将深度强化学习技术应用到了此类新问题的求解中。在之后的研究中将会深入探索更有效的网络模型和训练算法，并且会尝试将物品的摆放位置和朝向的选择整合到模型中。

附录：

A：三维装箱问题的启发式算法

用于求解本文的新型三维装箱问题的启发式算法包括最小表面积（least surface area）算法和最小浪费空间（least wastespace）算法。算法的详细步骤如下所示：

算法2：针对三维装箱问题的启发式算法

使用 I 表示物品集合，每一个物品的长宽高分别为 *li*、*wi* 和 *hi*

初始化一个足够大的箱子 B，其长宽高分别为 *L, W, H*（可以设置为 $L=W=H=\sum_{i=1}^{n} max(l_i,h_i,w_i)$ ）

初始化剩余物品集合 $\hat{I}=I$，初始化可用空间集合 $ES=\varnothing$

for t = 1 to n do

  如果 t =1：

    选择一个表面积最大的物品

    将物品放入箱子中，然后将由此产生的 3 个可用空间加入到 ES 中

  否则：

    使用最小浪费空间算法从物品集合 *S* 中选择一个物品 i，更新 $\hat{I} \leftarrow \hat{I} \setminus i$

    使用最小表面积算法选择物品的放入空间位置和朝向

    在放入物品之后，会产生新的可用空间 ($ES1$)，而之前的部分可用空间会被占用 ($ES2$)，更新可用空间集合 $ES \leftarrow ES \cup ES1 \setminus ES2$

  结束判断

end for

返回能够容纳所有物品的箱子的表面积

算法3：最小表面积算法

使用 *ES* 表示可用空间集合，*O* 表示物品的朝向集合
初始化物品 *i* 对应的最小化表面积 $LSA_i = 3 \cdot \left(max\left(l_i, w_i, h_i\right) \cdot n \right)^2$
初始化对于物品 *i* 的最佳可用空间 $\hat{s}_i = null$ ，最佳朝向 $\hat{o}_i = null$
for 任意 $s \in ES$ do
  for 任意 $o \in O$ do
    计算物品 *i* 以朝向 *o* 放入 *s* 之后的表面积 $SA_{i,s,o}$
    如果 $SA_{i,s,o} < LSA_i$ ：
      更新 $\hat{s}_i = s, \hat{o}_i = o, LSA_i = SA_{i,s,o}$
    否则，如果 $SA_{i,s,o} = LSA_i$ ：
      如果 $min\left(length\left(s\right) - o\left(l_i\right), width\left(s\right) - o\left(w_i\right), height\left(s\right) - o\left(h_i\right)\right)$ 小于 $min\left(length\left(s_i\right) - o\left(l_i\right), width\left(s_i\right) - o\left(w_i\right), height\left(s_i\right) - o\left(h_i\right)\right)$ ，则使用 s, o 更新 s_i, o_i .
    结束判断
  end for
end for
返回对于物品 *i* 的 $\hat{s}_i, \hat{o}_i$

算法 4：最小浪费空间算法

使用 I 表示剩余物品集合，使用 V_i 表示物品的体积
初始化最佳物品 $\hat{\mathrm{i}} = null$
初始化最小体积 $LV = \left(max\left(l_i, w_i, h_i\right) \cdot n \right)^3$
for 任意 $i \in I$ do
  计算物品 i 按照朝向 $\hat{o}_i$ 放入可用空间 $\hat{s}_i$ 之后（$\hat{s}_i, \hat{o}_i$ 通过最小表面积算法获得）的体积 *V*，使用 $LWV_i = V - V_i$ 表示最小浪费体积
  如果 $LWV_i < LV$ ：
    更新 $LV = LWV_i, \hat{\mathrm{i}} = i$
  结束判断
end for
返回物品 *i*

B：关于新型三维装箱问题为 NP-hard 问题的证明

引理 B.1: 本文提出的新型三维装箱问题为 NP-hard 问题。

证明：

首先，我们证明新型的二维装箱问题为 NP-hard 问题。为了完成此证明，我们将新型的二维装箱问题归约为一维的普通装箱问题。

对于一维的普通装箱问题，我们有 n 个物品，其尺寸分别为 $w_1, w_2, \cdots, w_n$，其中每一个 w_i 为正整数。箱子的容量为正整数 W。优化目标为最小化箱子的使用数量。

为了将新型的二维装箱问题归约为普通的一维装箱问题，我们假设有 n 个物品，其宽度分别为 $w_1, w_2, \cdots, w_n$，高度为 $1/\left(n \cdot max\left(w_i\right)\right)$。而且还有一个物品，其宽度为 W，高度为 $W \cdot n \cdot max\left(w_i\right)$，我们称此物品为基准物品。则可以定义一个新型的装箱问题为：找到一个能够装下所有 $n+1$ 个物品且表面积最小的箱子。

不失一般性，我们假设基准物品放在箱子的左下角。如果将一个其它物品放到基准物品的右方，则至少会使表面积增加 $\left(W \cdot n \cdot max\left(x_i\right)\right)/\left(n \cdot max\left(w_i\right)\right) = W$。而如果将其它物品放到基准物品的上方，则最多使表面积增加 W。所以所有的物品必须被放到基准物品的上方。

然后，我们证明物品在放置时没有进行旋转。如果在放置时有任意一个物品进行了旋转，则表面积至少增加 $W \cdot min\left(w_i\right)$。如果放置时没有物品进行旋转，则表面积至多增加 W，所以为了最小化表面积，物品放置时是没有进行旋转的。

所以，如果我们能够针对此种包含 $n+1$ 个物品的新型二维装箱问题找到一个最优解，则我们就能够同时获得对于普通一维装箱问题的最优解。即如果我们能够在多项式时间内求解此类新型二维装箱问题，则同样能够在多项式时间内求解以上的普通一维装箱问题。显然，这种情况不可能出现，除非 P=NP。

对于新型的三维装箱问题，我们可以同样在二维问题的基础上对每个物品增加一个长度 $1/\left(n \cdot max\left(w_i\right)\right)^2$。证明方法与以上相同。

号称史上最晦涩的算法 Paxos，如何变得平易近人？

阿里技术

阿里妹导读：Paxos（分布式一致性算法）作为分布式系统的基石，一直都是计算机系统工程领域的热门话题。Paxos 号称是最难理解的算法，其实真的这么困难么？

“X-Paxos”是阿里巴巴数据库团队面向高性能、全球部署以及阿里业务特征等需求，实现的一个高性能分布式强一致的 Paxos 独立基础库。X-Paxos 具体又有哪些优势，能给现有的系统带来什么收益呢？

背景

分布式一致性算法（Consensus Algorithm）是一个分布式计算领域的基础性问题，其最基本的功能是为了在多个进程之间对某个（某些）值达成一致（强一致）；进而解决分布式系统的可用性问题（高可用）。Paxos 是最重要的分布式一致性算法，很多人都把它作为“分布式一致性协议”的代名词（Mike Burrows, inventor of the Chubby service at Google, says that“there is only one consensus protocol, and that’s Paxos”）。

关于 Paxos 的历史和原理，已经有很多经典的论文可以参考，也有厂内外的文章有详细的描述，本文就不再重复了。感兴趣的同学可以阅读下这些论文 [1,2,3,4]。

业内

虽然 Paxos 的理论提出已经 17 年了，从第一个 Paxos 的工业实现到现在也已经 11 年了，但是直到近几年，无论是顶级会议，还是业内，Paxos 相关的文章和项目还是层出不穷。

转向业内，虽然使用了 Paxos 及各种变种的产品已经层出不穷；但是真正工业级的，独立的，Paxos 基础库还是相当的少见：Google 并没有开源其任何 Paxos 基础库（连包含 Paxos 的项目都没有开源过）；Facebook 也没有公布过包含 Paxos

的产品；Apache 有 Zookeeper，但是其协议并不能支持一个高吞吐的状态机复制，且并没有提供独立的第三方库，可供快速接入。

在 Github 上能找到的 Paxos 的独立库，star 最高的是腾讯于去年开源的 phxpaxos（后文会作为竞品进行详细对比）。

因此到目前为止业内还鲜有成熟可靠的，可供快速使用的独立第三方 Paxos 库，开源的 Paxos 生态也尚未成熟。

X-Paxos

愿景

我们的初衷并不是要做一个 Paxos 的公共库，X-Paxos 诞生于阿里巴巴的分布式数据库 AliSQL X-Cluster，但 X-Paxos 并不属于 AliSQL X-Cluster。Paxos 是分布式系统的基石，X-Paxos 可用于解决各种各样的分布式系统中的一致性问题。

因此在整个分布式数据库的设计之初，我们就独立设计了分布式一致性协议模块，并把它独立为 X-Paxos。X-Paxos 为 AliSQL X-Cluster 解决了分布式一致性问题，同样可以快速赋予其他系统分布式一致性能力。

分布式一致性需求，并不是 AliSQL X-Cluster 所特有的，很多系统都存在这高可用和强一致的需求，我们把 Paxos 的能力独立成一个基础库，希望能够把这个能力带给更多的其他系统。

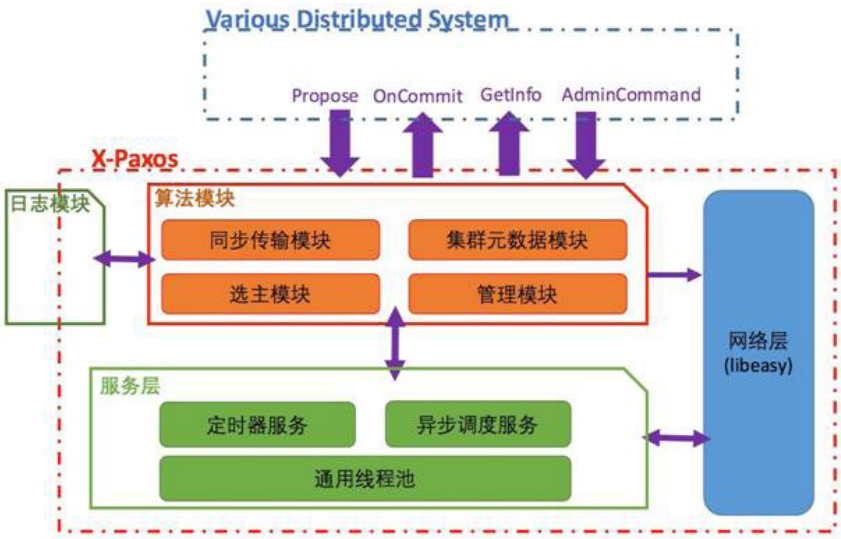
例如：团队内的同学把 X-Paxos 融入到单机 KV 数据库 RocksDB 中，快速实现了一个分布式 KV 引擎。集团已有业务团队团队把 X-Paxos 融入到业务存储系统中，构建全新的分布式强一致存储服务。

同时也正是 AliSQL X-Cluster，成就了 X-Paxos。Google 的论文《Paxos made live》中有一段话说的很好，大意是说：Paxos 从理论到现实世界的实现之间有巨大的鸿沟，在真正实现一个 Paxos 的时候，往往需要对 Paxos 的经典理论做一些扩展，（尤其是在实现一个高性能的 Paxos 的时候，扩展点就更多了，可以参考后文的功能增强和性能优化），这往往会导致真正的 Paxos 实现其实都是基于一个未被完全证明的协议。

这也就是传说中，理论证明一个 Paxos 的实现，比实现这个 Paxos 还要难的原因了。因此一个成熟的 Paxos 实现很难独立产生，往往需要和一个系统结合在一起，通过一个或者多个系统来验证其可靠性和完备性。这也是为什么大部分成熟的 Paxos 案例都是和分布式数据库相结合的，例如最早的 Paxos 实现（Chubby），当前的主要 Paxos 实现（Google 的 MegaStore、Spanner，AWS 的 DynamoDB、S3 等）。而 X-Paxos 正是依托于 AliSQL X-Cluster 验证了其可靠性和完备性。

我们的愿景是希望能够提供一个经过实践检验的，高度成熟可靠的独立 Paxos 基础库。使得一个后端服务能够通过简单的接入，就能拥有 Paxos 算法赋予的强一致、高可用、自动容灾等能力。真正将晦涩难懂的 Paxos，变得平易近人，带入千万家。

架构



X-Paxos 的整体架构如上图所示，主要可分为网络层、服务层、算法模块、日志模块 4 个部分

网络层

网络层基于阿里内部非常成熟的网络库 libeasy 实现。libeasy 的异步框架和线程池非常契合我们的整体异步化设计，同时我们对 libeasy 的重连等逻辑进行了修

改，以适应分布式协议的需求。

服务层

服务层是驱动整个 Paxos 运行的基础，为 Paxos 提供了事件驱动，定时回调等核心的运行功能，每一个 Paxos 实现都有一个与之紧密相关的驱动层，驱动层的架构与性能和稳定性密切相关。

X-Paxos 的服务层是一个基于 C++11 特性实现的多线程异步框架。常见的状态机 / 回调模型存在开发效率较低，可读性差等问题，一直被开发者所诟病；而协程又因其单线程的瓶颈，而使其应用场景受到限制。C++11 以后的新版本提供了完美转发（argument forwarding）、可变模板参数（variadic templates）等特性，为我们能够实现一种全新的异步调用模型提供了可能。

例如这是 X-Paxos 内实际的一行创建单次定时任务的代码

```
new ThreadTimer(srv_->getThreadTimerService(),srv_, electionTimeout_,
ThreadTimer::Oneshot,
&Paxos::checkLeaderTransfer, this, targetId, currentTerm_.load(),log_-
>getLastLogIndex());
```

以上一行程序，包含了定时器的创建，任意回调函数的设置，回调函数参数的转发，并保证在回调触发后（Oneshot）内存的自动回收。同时服务层支持嵌套回调，即在回调函数中再一次生成一个定时 / 即时任务，实现一个有限次定时循环逻辑。

算法模块

X-Paxos 当前的算法基于 unique proposer 的 multi-paxos[3] 实现，大量理论和实践已经证明了基于 unique proposer 的 multi-paxos，性能好于 multi-paxos/basic paxos，当前成熟的基于 Paxos 的系统，大部分都采用了这种方式。

算法模块的基础功能部分本文不再重复，感兴趣的同学可以参考相关论文 [1,2,4]。在基础算法的基础上，结合阿里业务的场景以及高性能和生态的需求，X-Paxos 做了很多的创新性的功能和性能的优化，使其相对于基础的 multi-paxos，功能变的更加丰富，在多种部署场景下性能都有明显的提升。下一章中，将对这些优化进行详细的介绍。

日志模块

日志模块本是算法模块的一部分，但是出于对极致性能要求的考虑，我们把日志模块独立出来，并实现了一个默认的高性能的日志模块；有极致性能以及成本需求的用户，可以结合已有的日志系统，对接日志模块接口，以获取更高的性能和更低的成本。这也是 X-Paxos 作为高性能独立库特有的优势，下一章会进行详细的介绍。

功能增强

结合广泛的业务场景，构建开放的生态

1. 在线添加 / 删除节点，在线转让 leader

X-Paxos 在标准 multi-paxos 的基础上，支持在线添加 / 删除多种角色的节点，支持在线快速将 leadership 节点转移到其他节点（有主选举）。

2. 策略化多数派和权重化选主

集团及蚂蚁目前的多地有中心的架构，很多应用因其部署的特点，往往要求其在未发生城市级容灾的情况下，仅在中心写入数据库，或调用其他分布式服务；同时又要求在发生城市级容灾的时候（同一个城市的多个机房全部不可用），可以完全不丢失任何数据的情况下，将写入点切换到非中心。

而经典的 multi-paxos 并不能满足这些需求。经典理论中，多数派强同步以后即可完成提交，而多数派是非特定的，并不能保证某个 / 某些节点一定能得到完整的数据，并激活服务。在实际实现中，往往地理位置较近的节点会拥有强一致的数据，而地理位置较远的节点，一直处于非强一致节点，在容灾的时候永远无法激活为主节点，形同虚设。

同时当中心单节点出现故障需要容灾的时候，往往需要将主节点就近切换到同中心的另外一个节点；由于应用在多地的部署往往是非对称的原因，才出现单个 region 全挂的时候，写需要将主节点切到特定的 region 内。这些需求都需要 Paxos 在选主的时候，可以由用户指定规则，而经典理论中同样没有类似的功能，添加权重也需要保证 Paxos 的正确性。

X-Paxos 在协议中实现了策略化多数派和权重化选主。**基于策略化多数派**，用

户可以通过动态配置，指定某个 / 某些节点必须保有强一致的数据，在出现容灾需求的时候，可以立即激活为主节点。**基于权重化选主**，用户可以指定各个节点的选主权重，只有在高权重的节点全部不可用的时候，才会激活低权重的节点。

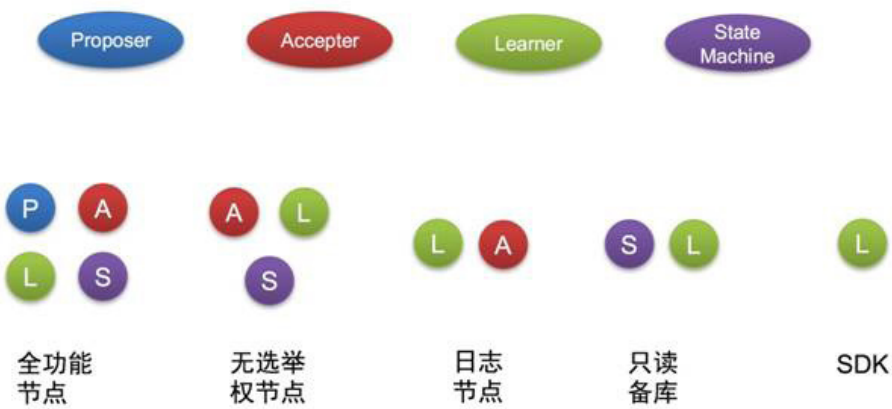
3. 节点角色定制化（Proposer/Accepter/Learner 的独立配置）

在经典的 multi-paxos 实现中，一般每个节点都包含了 Proposer/Accepter/Learner 三种功能，每一个节点都是全功能节点。但是某些情况下我们并不需要所有节点都拥有全部的功能，例如：

1. 经典的三个副本部署中，我们可以裁剪其中一个节点的状态机，只保留日志（无数据的纯日志节点，但是在同步中作为多数派计算），此时我们需要裁剪掉协议中的 Proposer 功能（被选举权），保留 Accepter 和 Learner 功能。

2. 我们希望可以有若干个节点可以作为下游，订阅 / 消费协议产生的日志流，而不作为集群的成员（不作为多数派计算，因为这些节点不保存日志流），此时我们裁剪掉协议的 Proposer/Accepter 功能，只保留 Learner 功能

当然还有其他的组合方式，通过对节点角色的定制化组合，我们可以开发出很多的定制功能节点，即节约了成本，又丰富了功能。



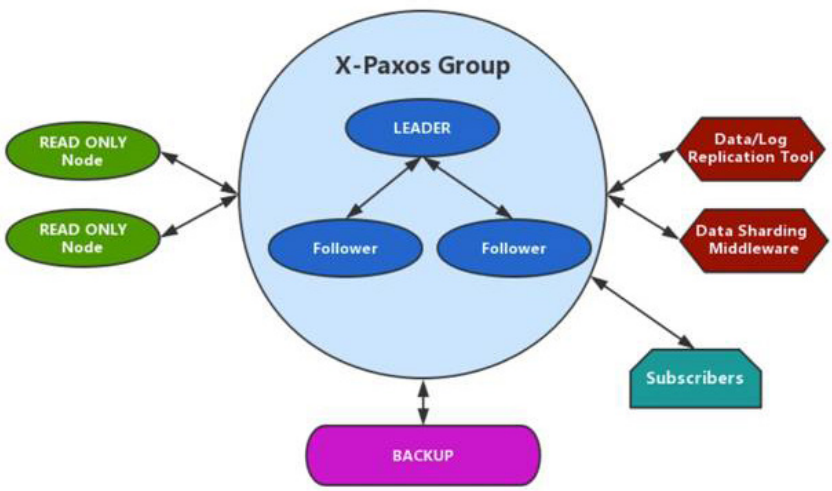
4. Witness SDK

基于上节节点角色定制化中的单独 Learner 角色的功能，引发了无穷的想象力。

Learner 角色，可以抽象成一个数据流订阅者（Witness Node），整个集群中可以加入无数个订阅者，当有新的日志被提交的时候，这些订阅者会收到其关心的日志流，基于订阅者功能，我们可以让一个集群很容易的实现下游订阅消费，日志即时备份，配置变更推送等等的功能。

因此我们把 Learner 角色单独封装成了一个 SDK。基于这个 SDK，用户可以快速的为自己的集群添加，订阅注册，流式订阅定功能；结合特定的用途打造一个完成的生态。

例如日志流 SDK 在 AliSQL X-Cluster 中打造的生态：



采用了 X-Paxos 也可以利用 Witness SDK 快速实现分布式系统和下游的其他系统的对接，形成一个完整的生态。

我们拿 MySQL 的日志（binlog）备份来举例：

- 普通方案
  - 每隔固定时间 Tb，将 MySQL 生成的 binlog 文件备份到永久备份系统（OSS、S3 等）
  - RPO (Recovery PointObjective) 为 Tb
- SDK 方案

- X-Paxos 支持由 SDK 订阅增量日志，备份系统只需要简单的实现从 SDK 流到 OSS 流的对接，即可实现流式备份
- RPO (Recovery PointObjective) 为 0 除备份以外，Witness SDK 在下游流式订阅（DRC）、自封闭高可用系统（X-Driver）、异步只读备库等方面都有实战案例，更多的应用案例在不断的添加中。

性能优化

我们一直坚信网络延迟不应该影响吞吐

1. Batching & Pipelining

Paxos 除了设计之初的强一致和高可用以外，其高性能也是至关重要的，尤其是应用于 AliSQL X-Cluster 这种高性能分布式数据库的时候，对协议的吞吐，延迟都提出了很高的要求。同时作为可全球部署的分布式一致性协议，在高延迟下的性能挑战变得尤为重要。

X-Paxos 针对高延迟网络做了大量的协议优化尝试和测试，并结合学术界现有的理论成果 [5,6,7] 通过合理的 Batching 和 Pipelining，设计并实现了一整套自适应的针对高延迟高吞吐和低延迟高吞吐网络的通信模式，极大的提升了 X-Paxos 的性能（对比见下节）。类似的优化在同类竞品中还非常的罕见。

Batching 是指，将多个日志合并成单个消息进行发送；Batching 可以有效的降低消息粒度带来的额外损耗，提升吞吐。但是过大 Batching 容易造成单请求的延迟过大，导致并发请求数过高，继而影响了吞吐和请求延迟。

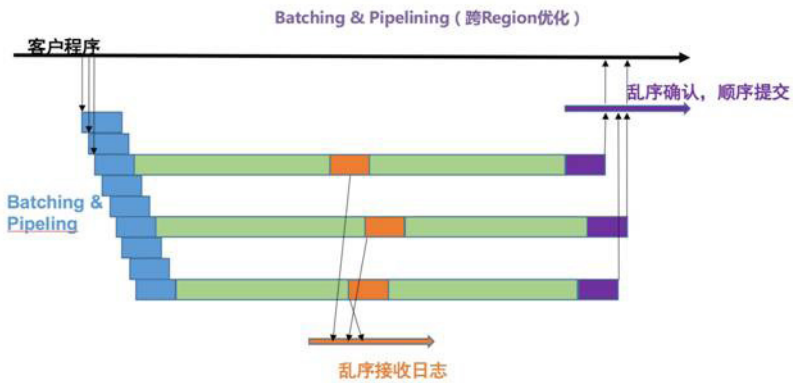
Pipelining 是指在上一个消息返回结果以前，并发的发送下一个消息到对应节点的机制，通过提高并发发送消息数量（Pipelining 数量），可以有效的降低并发单请求延迟，同时在 transmission delay 小于 propagationdelay 的时候（高延迟高吞吐网络），有效提升性能。

经推导可知 Batching（消息大小 :M）和 Pipeling（消息并发 :P）在如下关系下，达到最高吞吐

```
M/R * P = D
```

其中 R 为网络带宽，D 为网络传播延迟（propagation delay，约为 RTT/2）X-Paxos 结合以上理论，通过内置探测，针对不同节点的部署延迟，自适应的调整针对每个节点的 Batching 和 Pipeling 参数，达到整体的最大吞吐。

Pipeling 的引入，需要解决日志的乱序问题，特别是在异地场景下，window 加大，加大了乱序的概率。X-Paxos 实现了一个高效的乱序处理模块，可以对底层日志实现屏蔽乱序问题，实现高效的乱序日志存储。



2. 多线程，全异步的 Paxos 库

由于 Paxos 的内部状态复杂，实现高效的单实例多线程的 Paxos 变成一个非常大的挑战。无论我们上面提到的 github 中 star 最多的 phxpaxos，还是 Oracle MySQL Group Replication 中使用的 xcom，**都是单线程的实现**。**phxpaxos** 采用了单分区单线程，多实例聚合的方式提升总吞吐，但是对单分区的性能非常的有限；而 **xcom** 是一个基于协程的单线程实现。单线程的 Paxos 实现，在处理序列化 / 反序列化，分发、发包等逻辑的时候都为串行执行，性能瓶颈明显。

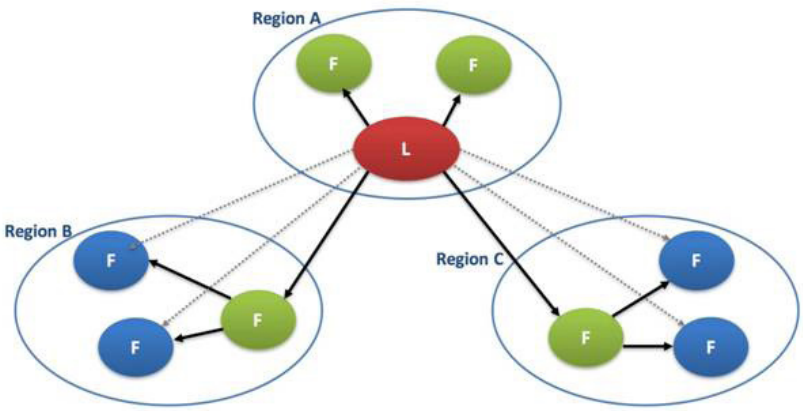
X-Paxos 完全基于多线程实现，可以在单个分区 Paxos 中完全的使用多线程的能力，所有的任务都有通用的 woker 来运行，消除了 CPU 的瓶颈。依赖于服务层的多线程异步框架和异步网络层，X-Paxos 除了必要的协议串行点外，大部分操作

都可以并发执行，并且部分协议串行点采用了无锁设计，可以有效利用多线程能力，实现了 Paxos 的单分区多线程能力，单分区性能远超竞品，甚至超过了竞品的多分区分区性能。

3. Locality Aware ContentDistribution

基于 unique proposer 的分布式系统的存在的一个瓶颈点就是主节点是唯一的内容输出源，当集群存在大量节点的时候，主节点的大量网络收发工作会导致主节点的负载过大，引发 CPU 和带宽的瓶颈；在全国 / 全球部署的时候，所有节点和主节点之间的直接通信会造成跨 region 之间的长传 / 国际链路的带宽占用过大。

X-Paxos 是旨在解决全球分布式强一致问题的 Paxos 独立库，在设计之初就考虑到了这个问题。X-Paxos 在稳态运行时会感知各个节点之间的网络延迟（物理距离），并形成级联拓扑，有效降低主节点的负载，降低长传链路的带宽使用；而在有节点异常的时候，又会自动重组拓扑，保证各个存活节点间的同行的正常进行。同时 X-Paxos 支持有业务来设定重组拓扑的规则，业务可以根据自己 APP 的部署架构和延迟特性来针对性的设置拓扑重组规则。



4. 可插拔日志

X-Paxos 和现有的大部分 paxos 库很大的不同点就是 X-Paxos 支持可插拔的日志模块。日志模块是 Paxos 中一个重要的模块，它的持久化关系到数据的一致性，

它的读写性能很大程度上会影响协议整体的读写性能。当前大部分独立 Paxos 库都是内置日志模块，并且不支持插拔的。这会带来 2 个弊端：

1. 默认的日志模块提供通用的功能，很难结合具体的系统做针对性的优化

2. 现有的系统往往已经存在了 WAL（Write Ahead Log），而 Paxos 协议中需要再存一份。这使得 a) 单次 commit 本地需要 sync 2 次（影响性能）；b) 双份日志浪费了大量的存储

例如：**phxpaxos** 内置的日志模块采用 LevelDB，作为日志系统其操作大量冗余，无针对优化，性能堪忧；同时采用 phxpaxos 的 phxsql 单节点需要即保存 binlog，又保存 Paxos log（在独立的 phxbinlogsvr 中），严重影响了性能，浪费了存储空间。而采用 **X-Paxos** 的 AliSQL X-Cluster 直接改造了现有的 binlog 模块，对接到 X-Paxos 的日志模块，单节点仅一份日志，即降低了存储，又提高了性能。

分布式正确性验证

对于一个分布式强一致协议来说，正确性是生命线。上文已经提及，一个分布式强一致协议，很难完整的理论证明其正确性，再加上工程实现的问题，困难就更多了。我们从理论和工程 2 方面用了大量的理论和技术手段来保证 X-Paxos 的正确性和完备性。

1. Jepsen

- Jepsen 是开源社区比较公认的分布式数据库的测试框架。Jepsen 验证过包 VoltDB、CockroachDB、Galera、MongoDB、etcd 在内的几乎所有的主流分布式数据库 / 系统，检验出了不少的问题。
- X-Paxos 完成了和 Jepsen 的对接，并验证了各个分布式数据库已有的 case。

2. TLA+

- TLA+ 是 Paxos 创始人、图灵奖获得者 Leslie Lamport 老爷子发明的一种形式化规约语言。TLA+ 专用于设计、建模和验证分布式并发系统。Amazon DynamoDB/S3/EBSMicrosoftCosmos DB 都通过 TLA+ 的模型验证发现了不少问题

- X-Paxos 目前已经通过了 TLA+ 的模型验证。

3. 随机异常系统

- 我们搭建了一套自动随机异常验证系统，可以自动化验证各种异常场景下的协议正确性和稳定性。验证 X-Paxos 在模拟网络丢包、闪断、隔离，磁盘故障等情况下的功能正确和数据一致。

4. 异常回归系统

- X-Paxos 拥有一套异常 case 回归系统，对于曾经出现过或者预期的并发异常 case，都会加到异常 case 库中，进行日常回归验证。同时异常 case 库也在不断的丰富中。

竞品分析和对比

XCOM (MySQL Group Replication)

MySQL GroupReplication 是 MySQL 官方借鉴 Galera 的思想，在 MySQL 上构建分布式强一致集群的一个插件。MySQL Group Replication 早期采用的分布式协议是 CoroSync，这是由 Red Hat 开发的基于 Totem（The Totem Single-Ring Ordering and MembershipProtocol）[8] 协议开发的分布式一致性协议库；由于 Totem 算法本身存在的一些局限性能原因，从 MySQL 5.7.9 以后，官方采用了自己研发的基于类 Paxos(Mencius)[10] 的一致性协议库 XCOM。

XCOM 是 MySQL Group Replication 的核心组件，称为 Group Communication Core[9]。我们分析了 XCOM 的源码，XCOM 内部是一个由纯 C 语言编译的核心模块以及有 C++ 实现的 proxy 实现的系统。纯 C 模块由单线程驱动，依赖协程实现任务调度。因此 Client(MySQL GroupReplication Plugin) 必须用 tcp 连接向 XCOM 发送请求。因此 XCOM 存在如下的不足之处：

1. 单线程驱动，无多线程能力

- 架构决定，很难突破

2. 通信流需要额外的一次 TCP 协议栈

- 在内存拷贝都要精细计算的应用中，线程间多一次网络通信很难接受

3. XCOM 虽然实现了 Batching 和 Pipelining，但是其值均为固定值，很难适应真实的场景

- 官方的文档中也提到了这一点 [9]
- 这也使得 MySQL Group Replication 在跨 Region 场景中性能很不理想（见 AliSQL X-Cluster 对比测试）

phxpaxos (phxsql)

phxpaxos 是腾讯推出的基于 Paxos 协议的独立库，其和 MySQL 结合后推出了 phxsql 项目，也是基于 MySQL 实现的分布式强一致 MySQL 集群。phxpaxos 可独立用于其他项目，是目前 github 上 star 最多（1000+）的 Paxos 独立库。关于 phxsql 的细节本文不再叙述，可以参考（AliSQL X-Cluster 的竞品分析部分），我们这里主要分析 phxpaxos。

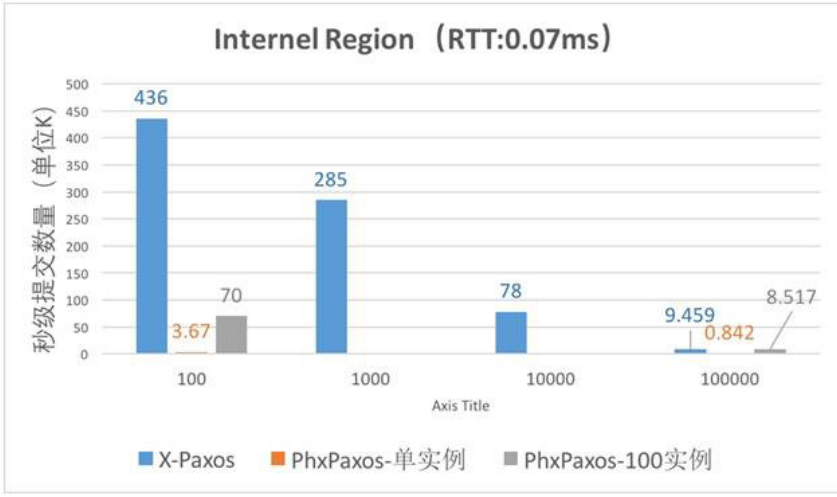
phxpaxos 也是基于 multi-Paxos 实现的独立库，其架构上采用单 Paxos 单线程设计，但是支持多 Paxos 分区以扩展多线程能力，这种扩展需要多数据进行提前分区。因此 phxpaxos 的不足之处如下：

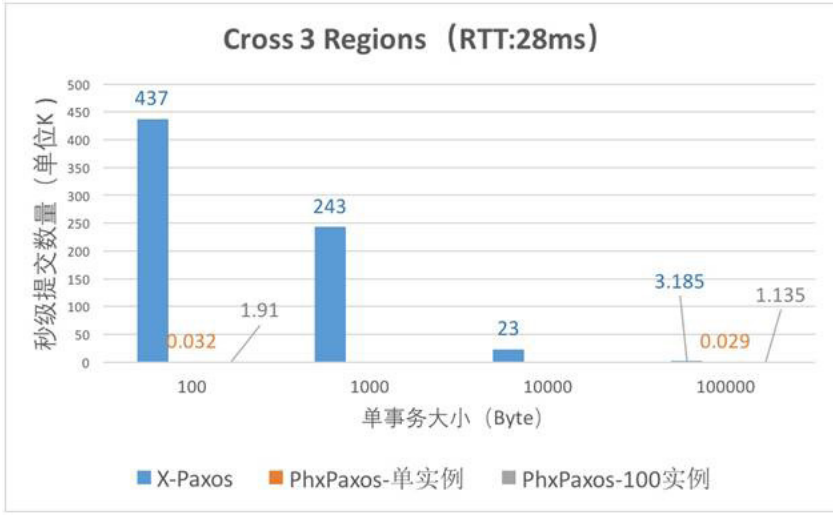
1. 单 Paxos 对象只支持单线程；可支持多 Paxos 对象，共享网络层
2. 不支持 pipelining，在跨 Region 环境（高延迟）下，几乎不可用
3. 多份日志冗余，基于 LevelDB 的日子系统性能瓶颈

性能对比

我们还是拿腾讯的 phxpaxos 作为竞品和我们进行对比（XCOM 无独立组件，可间接参考 MySQL Group Replication 和 AliSQL X-Cluster 的对比测试结果）

我们分别在 a) Region 内 3 节点部署 b) 3 个 Region 各一个节点部署调节下，以不同的请求大小进行压测。





从上面 2 个对比图中可以看到：

1. X-Paxos 的同 Region 性能是 phxpaxos 的 100 倍以上

2. 跨 Region 场景下 phxpaxos 几乎不可用，而 X-Paxos 在 444Byte（sysbench insert 场景单请求大小），性能只有 3.5% 的下降，几乎不影响吞吐。

现状与未来

现状

目前 X-Paxos 一期已经发布上线。

基于 X-Paxos 的集团数据库团队产品 AliSQL X-Cluster 已在集团内广泛使用。

X-Paxos 和业务系统结合打造的分布式服务也相继落地上线。

未来

Paxos 是分布式系统的基石，即使是近几年，学术界关于 Paxos 的文章，新的演进方向一致在不断的涌现，我们的 X-Paxos 也会不停的发展，以更好的适应集团内外的需求，未来主要的发展方向如下：

- 高效率，多分区支持
  - 基于新的异步框架，实现一个深度底层共享的多分区 Paxos
- 多节点强一致读
  - 经典的 multi-paxos 只有在 leader 上才能提供强一致读，spanner 和业界都有一些在多节点上提供强一致读的方案，但还是有比较明显的缺陷。

最后，无论是对 Paxos 协议感兴趣的同学，还是想更多的了解或使用 X-Paxos 的同学，都可以在微博联系我（@ 小强 -zju），欢迎一起交流，互相学习！

参考资料

[1] The part-time parliament
[2] The Chubby lock service forloosely-coupled distributed systems
[3] Paxos Made Simple
[4] Paxos Made Live - AnEngineering Perspective
[5] Everything You Ever Wantedto Know About Message Latency
[6] Adaptive Batching forReplicated Servers
[7] Tuning Paxos forhigh-throughput with batching and pipelining
[8] The Totem single-ringordering and membership protocol
[9] Group Replication: AJourney to the Group Communication Core
[10] Mencius: Building EfficientReplicated State Machines for WANs

在线视频衣物精确检索技术

方广　磐君　思淘

阿里妹导读：CVPR 是由全球最大的非营利专业技术学会 IEEE（电气和电子工程师协会）举办的计算机视觉领域的国际顶会，2017CVPR 收到超过 2500 篇论文投递，最终收录不到 800 篇，阿里巴巴集团 iDST 和 AI LAB 有多篇论文被收录。

今天为大家深入解读被 CVPR 2017 收录的论文之一、来自阿里巴巴 iDST 视频分析团队的《从视频到电商：视频衣物精确检索》。

《从视频到电商：视频衣物精确检索》围绕视频电商业务场景，提出了一个在线视频衣物精确检索系统。该系统能够满足用户在观看影视剧时想要同时购买明星同款的需求。

整个系统采用了目前最先进的衣物检测和跟踪技术。针对明星同款检索中存在的多角度、多场景、遮挡等问题，提出可变化的深度树形结构（ReconfigurableDeep Tree structure）利用多帧之间的相似匹配解决单一帧检索存在的遮挡、模糊等问题。该结构可以认为是对现有 attention 模型的一种扩展，可以用来解决多模型融合问题。

论文技术在天猫魔盒视频中应用

业务场景及研究问题：视频电商中的衣物精确匹配

早在 2014 年，阿里与优酷土豆发布视频电商战略，称未来可以实现边看边买，使得视频电商的概念，继微博电商，朋友圈电商之后浮出水面。电商平台拥有少量商品，而视频网站具有巨大的流量，二者结合是发展的必然结果。电商平台可以借助视频网站的流量来实现导流和平台下沉，而视频网站则需要通过广告点击和商品成交来实现流量变现，因此二者的结合可谓一拍即合。

视频电商的商业主旨是打造以视频为入口的购物服务，视频中出现所有物体都可能是商品，提供包括边看边买、明星同款、广告投放等服务，它集娱乐、休闲、购物于一体，给用户构造出一种“身临其境”情境营销，或者是明星同款的冲动式消费。视频电商目前已经不是停留在概念层次了，视频网站向电商的导流转化也一直在不断的尝试中。

影视剧中的服饰存在较大的差异性和异构性，同一个目标往往展现出较大的差异。服饰购物图像通常具有杂乱、多样的背景，而且常在户外拍摄。多样化的背景可能是建筑物，街道、风景、汽车等多种情况。由于自然场景下受到光线、角度、大小、分辨率、几何学和光度学的变化等影响，使得服饰呈现出现的外形极为复杂，即使是同一件服饰也会出现变化较大的效果。

同时在线网站为更好地展示服饰的效果，通常聘请时尚模特穿着所售商品，模特 / 人物姿势变化也是导致服饰变化的一个重要因素。由于以上这些因素，使得视频明星同款搜索成为了一个极具挑战性的技术问题。

网络结构及技术细节

AsymNet 网络结构：整个 Asymnet 深度神经网络结构如图 1 所示。当用户通过机顶盒（天猫魔盒）观看视频时，该网络将从电商网站（淘宝、天猫）检索到与之匹配的衣服，并推荐给用户。

为忽略复杂背景对检索结果的影响，更准确的进行服装定位，我们首先应用服饰检测技术，提取得到服饰区域一组候选框。然后对这些候选框进行跟踪，得到明星同款在视频中的的运动轨迹。对于衣物候选区域和运动轨迹我们分别利用用图像特征网络（IFN）和视频特征网络（VFN）进行特征学习。

考虑到服装的运动轨迹，衣物精确检索问题被定义为不对称（多对单）匹配问题，我们提出可变化的深度树形结（Reconfigurable Deep Tree Structure），利用多帧之间的相似匹配解决单一帧检索存在的遮挡、模糊等问题。后续本文将详细介绍模型的各个部分。

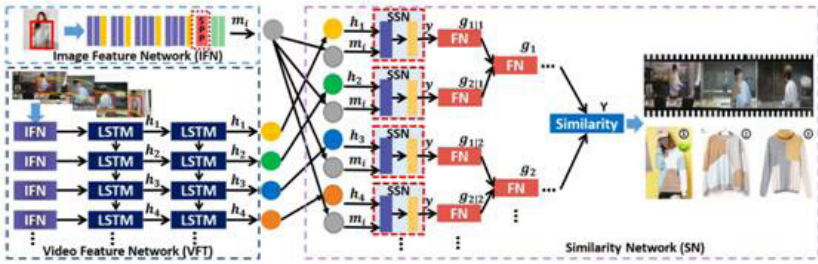


图 1　Asymnet 深度神经网络结构

图像特征网络（IFN）：传统 CNN 网络要求输入图像为固定的 227x227（因为 CNN 网络中的卷积层需要有一个确定的预定义的维度）。在视频电商业务场景中，因为衣物检测候选框为任意大小，尺度变化很大，传统 CNN 网络无法进行有效的特征学习。

针对这一问题，我们利用空间金字塔池化结构（SPP）体系结构，如图 2 所示。它通过空间池聚合最后一个卷积层的特征，从而使池区域的大小与输入的大小无关。

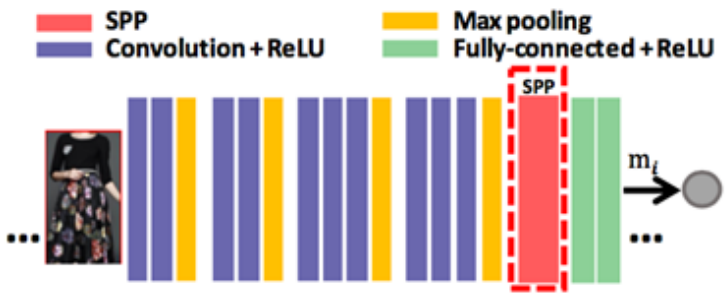


图 2　Asymnet 图像特征网络（IFN）

视频特征网络（VFN）：为了更好的考虑视频的空间序列模式，进一步提高衣物检索的性能。基于 LSTM，我们提出了视频特征网络 (VFN)，如图 3 所示。其中实验验证明两层堆叠式 LSTM 结构能够在视频特征学习中得到最佳性能。

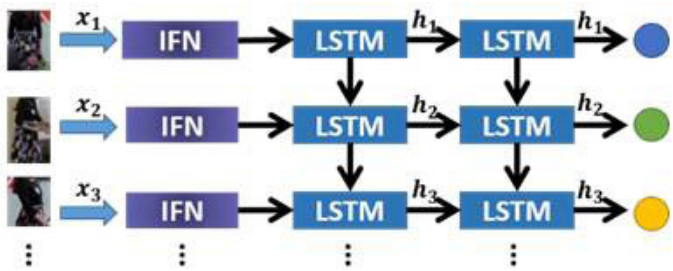


图 3　Asymnet 视频特征网络 (VFN)

相似性网络：明星同款匹配不同于近似衣物检索，精确匹配要求完全一致。在完全一致的要求下，传统的通过相似性计算来进行检索的方法，不能满足明星同款精确匹配要求。已有的方法通常将精确匹配问题转换为一个二分类问题，但这种方式适应性差，只能利用单一时刻的视频帧。

为了能够利用整个衣物运动轨迹，我们提出了如下的可变化的深度树形结构（ReconfigurableDeep Tree structure）将匹配问题转换为逻辑回归问题。匹配网络拟采用基于混合专家系统的逻辑回归网络。该结构可以认为是对现有 attention 模型的一种扩展，可以用来解决多模型融合问题。

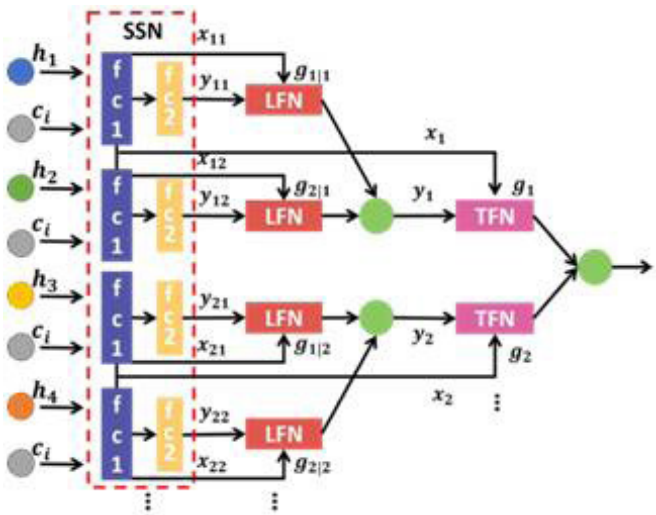


图 4　Asymnet 相似性网络

整个模型的目标函数是综合考虑每一帧的匹配结果，得到基于整个衣物运动序列和电商衣物的相似性，整个系统可以建模为对如下目标公式进行求解：

$$P(Y|X,\theta)=\sum_j g_j(X,\mathbf{v_j})\sum_i g_{j|i}(X,\mathbf{v_{ij}})p(y_{ij}|X,W_{ij})$$

类似于 attention 机制，我们提出如下后验概率模型，来对上式进行求解：

$$h_j=\frac{g_j\sum_i g_{j|i}P_{ij}(y)}{\sum_j g_i\sum_i g_{j|i}P_{ij}(y)}$$

and

$$h_{ij}=\frac{g_{j|i}P_{ij}(y)}{\sum_j g_{j|i}P_{ij}(y)}$$

得到如下梯度并采用端到端方式进行网络学习。

$$\nabla \mathbf{v_j}=\alpha\sum_t(h_j^{(t)}-g_j^{(t)})\mathbf{x_j}^{(t)} \qquad (10)$$

$$\nabla \mathbf{v_{ij}}=\alpha\sum_t h_i^{(t)}(h_{j|i}^{(t)}-g_{j|i}^{(t)})\mathbf{x_{ij}}^{(t)} \qquad (11)$$

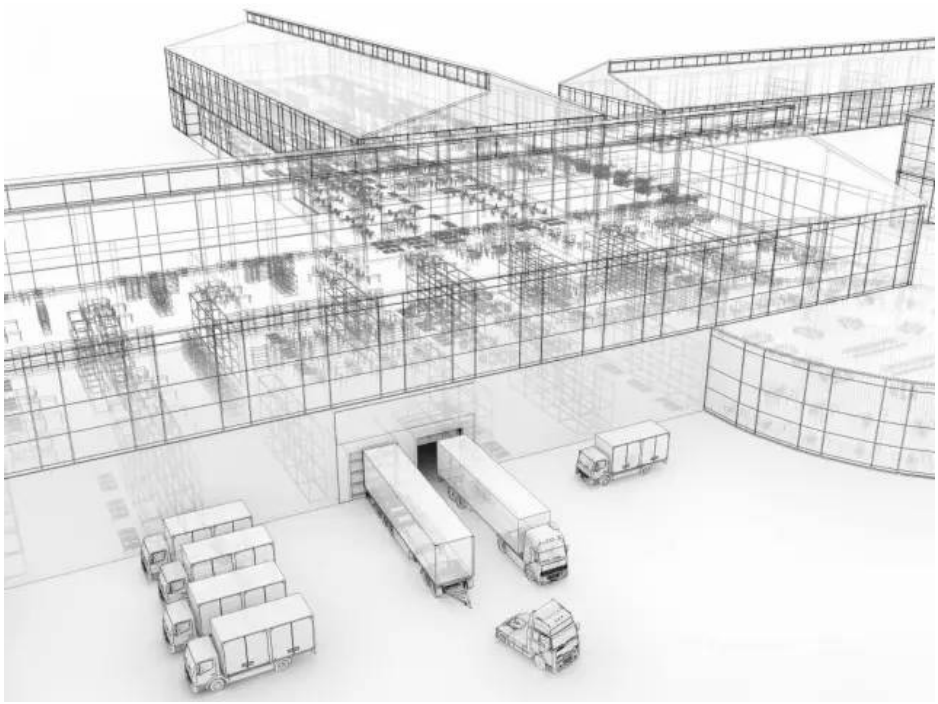
试验结果：我们利用业务数据和最新的衣物检索方法进行了对比，试验结果如下表所示。相对于 alexnet，Asymnet 在前 20 的检索精确率指标上，其性能几乎提高了进一倍。

相对于其他 2 种网络 CS 和 RC，我们发现 RC 的性能略优于 CS，因为 RC 具有较强的识别能力差异较小（采用多任务学习）。甚至在对于某些类别（无明显差别）RC 在精确率上甚至略好于 AsymNet，但是总的来说 AsymNet 比目前现有的方法拥有更好的性能。因为 Asymnet 可以处理现有的视频的时空动态变化，并结合自动视频帧的自动调节炉排判别信息的融合策略。

Category	# I	# TJ	# Q	# R	AL [15]	DS [8]	FT [14]	CS [1]	RC [23]	AsymNet
Outwear	18,144	5,581	1,116	3,628	17.31	22.94	26.97	27.61	31.80	**42.58**
Dress	14,128	4,346	869	2,825	22.93	24.90	25.56	29.33	34.34	**49.58**
Top	7,155	2,201	440	1,431	17.45	24.83	25.26	29.14	32.94	**35.12**
Mini skirt	6,571	2,021	404	1,314	23.35	24.83	27.47	29.50	31.30	**32.48**
Hat	6,534	2,010	402	1,306	15.82	13.98	20.19	25.87	33.81	**35.12**
Sunglass	6,133	1,886	377	1,226	11.85	7.46	11.35	11.83	**12.26**	12.16
Bag	5,257	1,617	323	1,051	23.78	27.63	27.47	25.67	25.48	**36.82**
Skirt	4,453	1,370	274	890	19.79	25.06	22.44	24.50	24.43	**41.75**
Suit	3,906	1,201	240	781	18.65	25.18	19.72	25.29	26.60	**42.08**
Shoes	3,358	1,033	206	671	11.45	24.10	23.92	25.03	**27.58**	26.95
Shorts	3,249	999	199	649	11.15	5.99	13.90	14.84	**16.62**	13.74
Pants	2,738	842	168	547	17.57	22.54	25.77	29.49	28.36	**32.13**
Breeches	2,044	628	125	408	23.45	22.99	25.03	28.52	28.76	**48.28**
High shoots	2,007	617	123	401	12.05	13.11	14.57	15.46	**16.04**	14.94
Overall	85,677	26,352	5,266	17,128	18.36	21.44	23.47	25.73	28.73	**36.63**

如何送货最省钱？菜鸟自研核心引擎架构解析

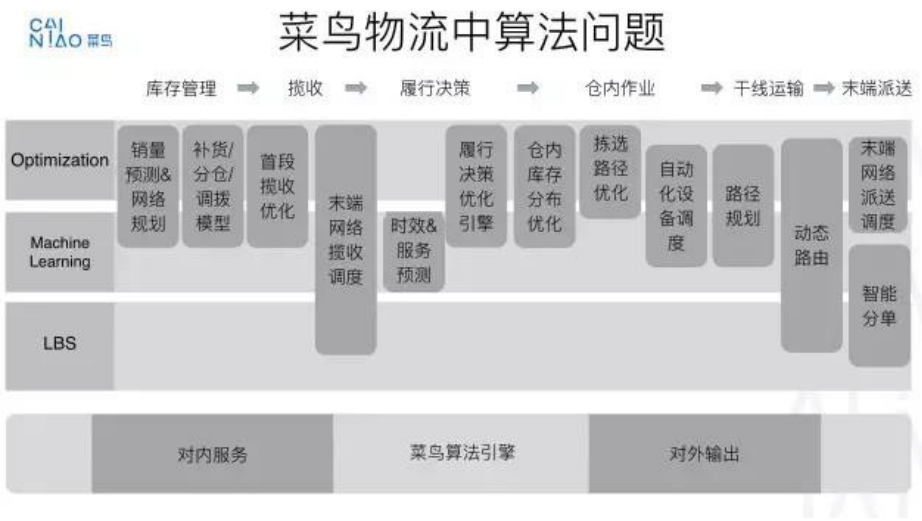
□ 本华



随着中国物流运输行业的蓬勃发展，物流成本已经占据了 18% 的国民生产总值，其中，车辆运输作为配送成本的核心要素，成为了一个必须面对的问题。而车辆路径规划问题的目标就是减少配送的车辆数目和距离，进而降低物流成本，同时也是物流成本透明化的重要手段。

菜鸟网络人工智能部从自身业务出发，联合集团 IDST、阿里巴巴云计算的力量，打造一款适合中国复杂的业务需求，又在效果上接近国际水准的分布式车辆路径规划求解引擎—— STARK VRP，以此向财富自由还继续追求黑科技的钢铁侠致敬。

菜鸟业务总览



由上图可见，车辆路径规划在整个链路中起到了举足轻重的作用。

运筹优化机器学习人工智能

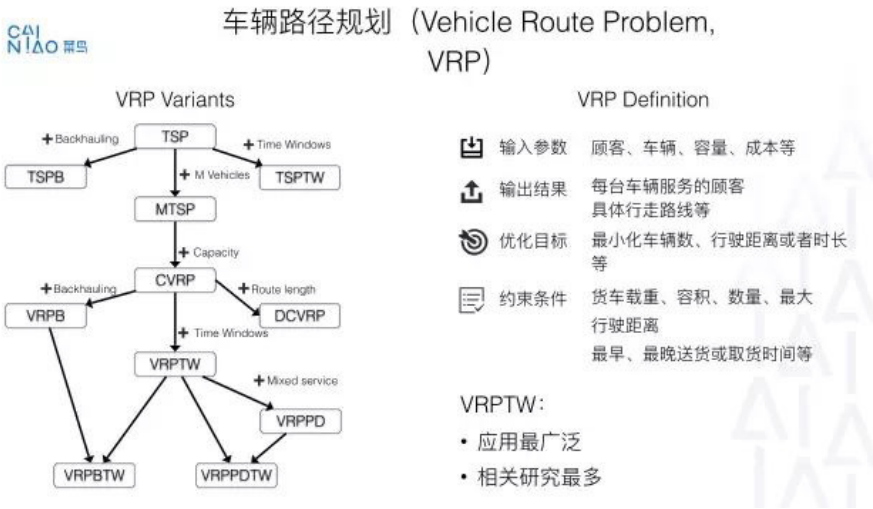
Nothing at all takes place in the Universe in which some rule of maximum or minimum does not appear.

——Leonhard Euler

作为优化领域历史悠久的问题，车辆路径规划问题已经被研究了数十年，我们从菜鸟自身的技术背景出发，充分利用自身庞大的计算资源为优势，探索一条结合运筹优化、分布式计算、机器学习、人工智能结合的技术路线。

问题定义

VRP 问题目标，是给出一个确定的最优解，包含车辆以及他们的运输路径，来服务一个客户集合的订单。 这也是组合优化中研究最广，最重要的问题之一。



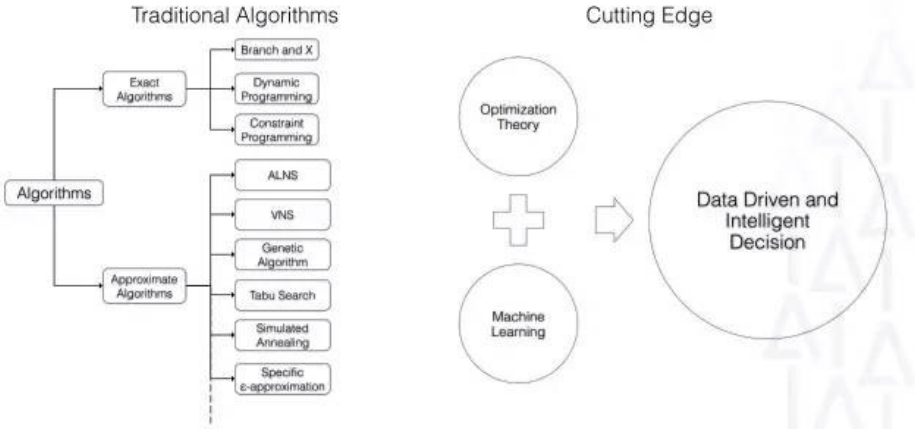
如大家所知，中国的物流情况尤为复杂，有自己很多独特的场景，也衍生出了对应的 VRP 求解类型和分支。以下是 STARK VRP 现阶段支持以及开发中的 VRP 类型和对应的业务类型。

- CVRP: Capacitated VRP, 限制车的体积、重量、客户数、最长距离等
- VRPTW: VRP with Time Windows, 针对客户有要求送达时间的场景，时间窗可以是多个
- VRPPD: VRP with Pickup and Delivery，外卖 O2O，快递员从不同的商店取货，送到不同的客户
- MDVRP：Multi-Depot VRP，同样的货物在多个仓库都可以获取，每个客户选择最佳的仓库
- OVRP：Open VRP，外包的私家车，在完成配送任务后，不需要返回仓库
- VRPB: VRP with backhaul，回程取货，回收返修的电子元器件
- Heterogeneous Fleet: 支持多车型，尤其适合中国目前配送资源是外包的情况
- T + n 时效：针对时效要求不高的，可以动态决定哪天送达，合并多日订单，减少车辆数

- Milk Run：同一辆车会循环取货
- Skilled VRP：某些客户只能由指定的车辆来服务，在中国司机会和客户之前形成一定的默契关系
- Same Route VRP：某些订单必须在一条路径上
- Generalized VRP：某个订单，有若干个 location，可从任一个取货，均可满足要求
- Split Delivery：某个客户的需求（当超过一辆车的容量时），可以由多辆车来分别送达
- Generalized VRP：某个订单，有若干个 location，可从任一个取货，均可满足要求
- VRP with intermediate facilities：针对新能源车的场景，考虑沿途的充电点以及载重量和耗电的关系
- 2E VRP：多级 VRP，适用于需要在不同的运输环节更换运输工具的场景，例如使用重卡运输到镇点之后，使用面包车或者无人机运输到村点

技术选型 – 丰富多样的求解方式

车辆路径规划（Vehicle Route Problem, VRP）



传统用于求解 VRP 的精确解法无法应对大规模数据集

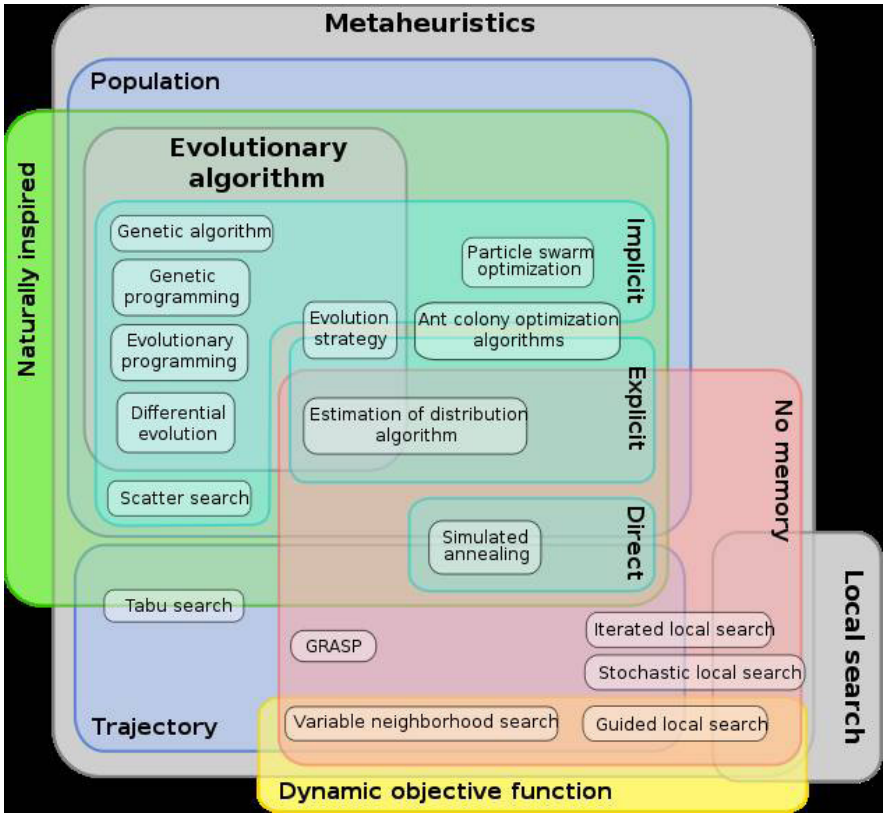
车辆路径规划（Vehicle Route Problem, VRP）

Exact Algorithm V.S. Heuristic Algorithm

Job数量		Gurobi	ALNS算法	分布式算法	ALNS/Gurobi	分布式算法/Gurobi
25 Jobs	车辆数	8	8	8	1.00	1.00
	距离	618.33	618.33	618.33	1.00	1.00
	计算时间(秒)	0.24	1.99	7.09	8.29	24.54
100 Jobs	车辆数	20	19	19	0.95	0.95
	距离	1642.88	1676.65	1650.8	1.02	1.00
	计算时间(秒)	50.46	5.72	29.08	0.11	0.58
200 Jobs	车辆数	24	21	20	0.88	0.83
	距离	4698.11	4986.31	4905.61	1.06	1.04
	计算时间(秒)	3600	21.80	415.54	0.01	0.12

在处理100个以上的Job时，精确解算法已经无法在合理的时间内给出令人满意的解。

利用元启发式构建求解的基础框架



在整个 VRP 算法迭代的过程中，我们顺势建立了一整套元启发式的框架，目前可以调用的包括：

- Large Neighborhood Search
- Adapative Large Neighborhood Search
- Variable Neighborhood Search
- Metaheuristic Hybrids
- Iterated Local Search
- Memetic Algorithm

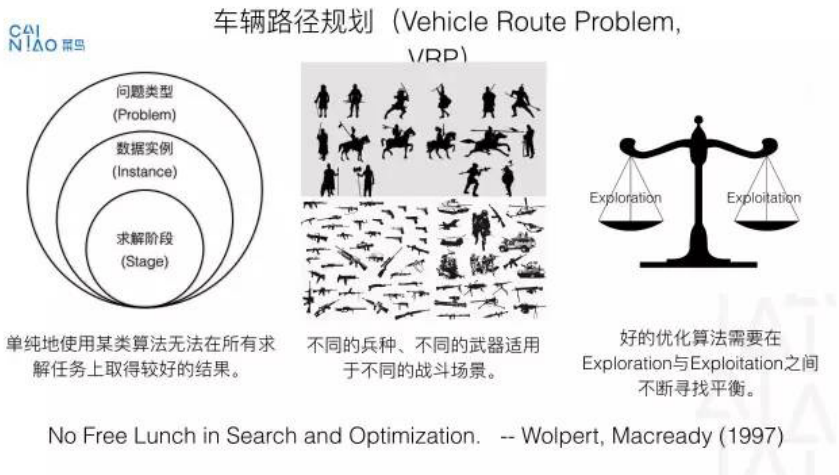
- Tabu Search
- Simulated Annealing
- Guided Local Search
- Fast Local Search

ALNS – Adapative Large Neighborhood Search

使用大规模领域搜索使得在每次迭代寻找一个更好的候选解集成为可能，并且能够指向一个更为有前途的搜索方向。

在实际过程中，不同的问题，甚至问题的不同阶段，每个 operator 的适用性和效果都是不同的，大家可以想象成在作战过程中，骑兵和坦克适用于大规模冲锋，但是在山路崎岖的地方就会行进艰难，而面对河流就直接无法通行。

属于 Hyper heuristics 的 ALNS 就是为了解决这一问题，它使用使用了 BANDIT 算法，根据每一次迭代的效果差异来确定下一次迭代各个算子的选择概率。

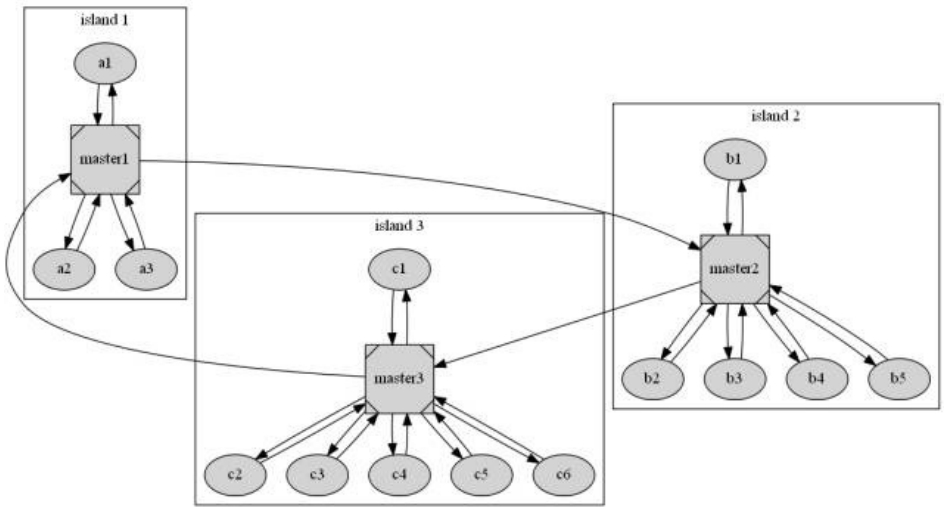


利用并行化提升效果

在效果的提升上，并行化是我们的重点方向之一，如果充分利用阿里在云计算和并行化的优势，是我们效果提升的关键。

ISLAND

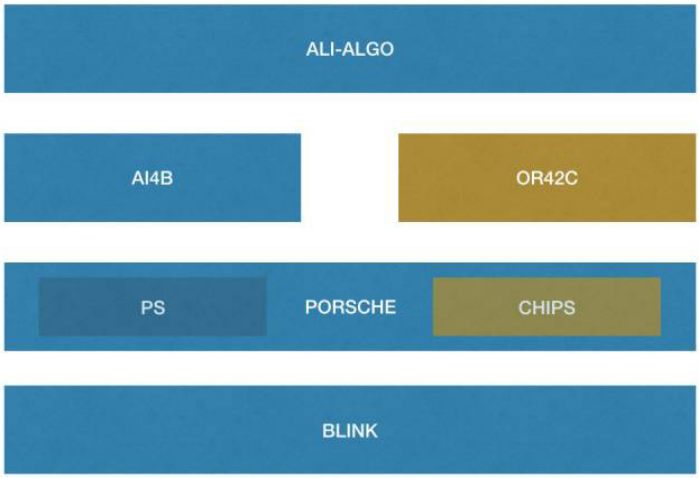
基于 ISLAND 的并行化思路，在于 island 之间以一定的机制动态发送和接受结果，保障搜索方向的有效性和利用多样性避免陷入 Local Minima。



EE Pool

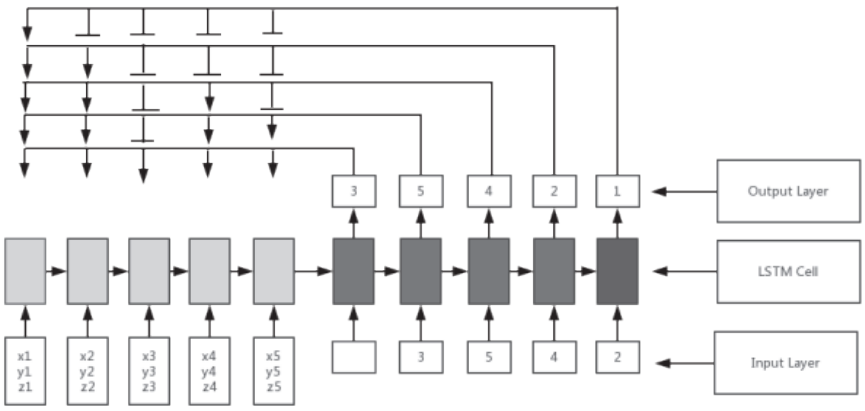
EE Pool 的思路是有一个核心的控制环节，在 island 之间通信的时候平衡 solution pool 的 exploration 和 exploitation，在不同的阶段调整追求 intensification 和 diversity 的平衡。整个控制过程采用 SSP，即不会在任何环节同步。

灵活的分布式架构



利用深度增强学习提升效果

这个方向是我们目前重点探索的方向之一，通过以某种 embedding 的方式表达 Problem，根据 Reinforcement Learning 的反馈，更新算子选择的概率，以期望在效率和效果获得提升，走向 data-driven。



业务效果的提升

村淘业务减少了 28% 的行驶距离

指标	原调度版本	STARK 版本
车辆数	108	52
距离	9158	6297

零售通业务减少 10% 的车辆

指标	原版本	STARK 版本
车辆数	22	20
距离	3000 公里	2600 公里

持平 6 项 Best Know Solution

Gehring & Homberger benchmark 保存了全球范围内有史以来已知的最好结果（Best Known Solution）

STARK VRP 在 400 Job 上持平了 4 项 BKS，在 1000 Job 上持平了 2 项 BKS。

	平均gap to bks （车辆数）	平均gap to bks(距离)	绝对持平	车辆数持平
400 Job	2.62%	1.45%	4	42
1000 Job	2.78%	6.53%	2	27

总结

我们希望通过以上几个实例让大家感受到车辆路径规划技术的重要性，这是有别于传统的基于机器学习的搜索、推荐、广告的 AI 赋能的另一种表达，它在日益快速发展的物流领域占据了不可或缺的一席之地，在无人驾驶大行其道的未来，它也是处于核心位置的调度中心。

STARK VRP 不仅仅在菜鸟内部的村淘、零售通、跨境、新能源车、仓内路径规划已经开始落地，而且更为广泛的开始服务于像日日顺、云鸟这样的外部公司，为降低中国的物流成本，提升时效尽一份算法人员的能力。

人类与机器人，如何能像朋友一样愉快聊天?

千诀

阿里妹导读：今天由阿里巴巴资深专家千诀带来精彩分享，主要聚焦在人和设备如何通过自然语言对话来展开对话交互。看完后你会发现，原来为了与你愉快聊天，机器人在背后付出了这么巨大的努力。

互联网正在从连接的时代走向交互的时代

纵观传统互联网时代，如果用一个词来总结和概括的话，“连接”这词再合适不过了，传统互联网时代，我认为主要建立了三种连接：第一，人和信息的连接；第二，人和人的连接；第三，人与商品服务的连接。第一种连接成就了 Google 和百度这样的互联网巨头；人和人的连接成就了 Facebook 和腾讯这样的互联网公司，人和商品服务的连接，成就了 Amazon、阿里巴巴、京东这样的巨头。所以，从这个意义上看，传统互联网最典型的特征就是连接。

过去 3-4 年，我们可以看到，连接互联网的设备发生了很大变化，设备已经从 PC 和智能手机延伸到更广泛的智能设备，比如智能音箱、智能电视、机器人、智能汽车等设备。智能设备的快速发展正在改变着人和设备之间的交互方式。

我们梳理一下智能设备，大致可以分成三类：

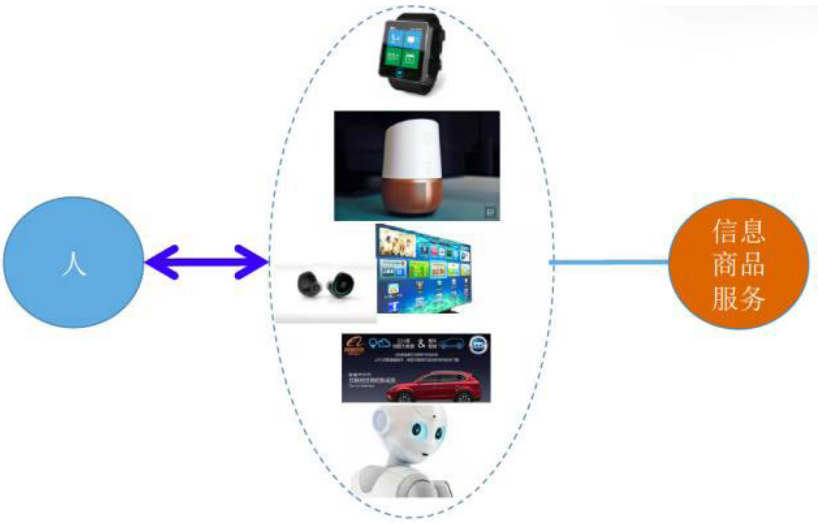
第一，可穿戴设备，比如说智能眼镜、智能手表。 第二，智能家居，比如我们熟知的智能电视、智能音箱、智能机器人、智能玩具。 第三，智能出行，比如智能汽车、智能后视镜、智能行车记录仪等。

我们从以下的一些数据来看智能设备的发展，这是美国市场做的调查：2015 年的时候，美国市场上语音设备的出货量是 170 万台，到 2016 年的时候这个数据上升到了 650 万台，大概是 3-4 倍的增速。专家预估，在 2017 年，这个数据将会上升到 2450 万台。

我们再来看用户，美国这边对使用智能助理的用户做了分析，把用户分成了三类：第一类，婴儿潮出生的用户，即 1945-1964 年出生的；第二类，1965-1980

年出生的；第三，80 后、90 后这批用户。这个调查发现，80 后、90 后对于智能助理产品的接受度，远远高于其他两类用户，比如说，2016 年，80 后 90 后的用户数大概在 2300 万，其他两个加到一起是 2000 万多一点。2017 年，预估的话大概是 3000 万，2018 年大概是 3500 万，到 2019 年是 4000 万。

所以，无论是用户的接受度，还是智能设备的快速发展和普及，都在促使人和设备之间交互方式的巨大改变，我称之为交互的时代。



今天的分享主要聚焦人和设备如何通过自然语言对话来展开对话交互的。

对话交互的特点，我认为主要有以下四点：

第一，人和智能设备和机器对话的交互一定是自然语言，因为对于人来说，自然语言是最自然的方式，也是门槛最低的方式。

第二，人和设备的对话交互应该是双向的。即不仅是人和设备说话，而且设备也可以和人对话，甚至在某些特定条件下机器人可以主动发起对话。

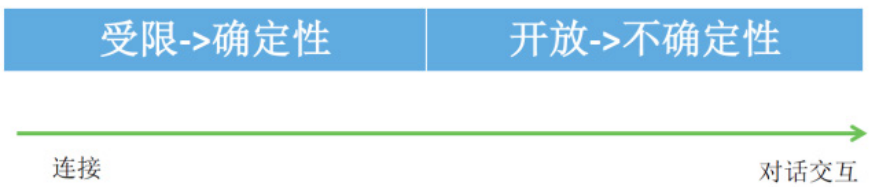
第三，人和设备的对话交互是多轮的，为了完成一个任务，比如定机票，这里会涉及多轮交互。

第四，上下文的理解，这是对话交互和传统的搜索引擎最大的不同之处，传统搜索是关键词，前后的关键词是没有任何关系的。对话交互实际上是要考虑到上下文，

在当前的上下文理解这句话什么意思。

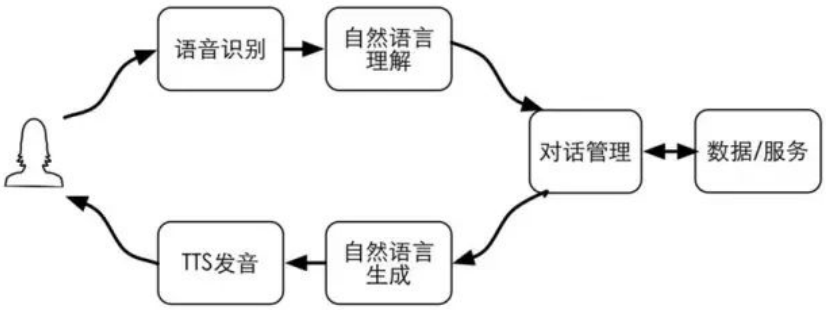
从连接到对话交互，一个本质的改变是什么？举个例子，比如淘宝网首页，抛开内容，其本质就是链接和按钮。对于用户来说，无论是点击链接还是按钮，他的行为完全是由产品经理定义好的，他的行为是完全确定的，所以，我认为它是一种受控、受限的行为，这种方式才能确保比较好的用户体验。再来看这种对话交互，用户可以说任何内容，天文、地理，包罗万象，这样一个完全开放的场景，机器人对自然语言的理解和把握对机器人来说是非常非常大的挑战，从而带来机器人回答的不确定性。

所以我认为从连接到对话交互这背后的本质改变是从确定性变为不确定性。那么后面无论是算法还是交互设计，基本上都要想办法提高语言理解的确定性或者是降低交互的不确定性。



阿里巴巴在智能对话交互方向上的进展和实践

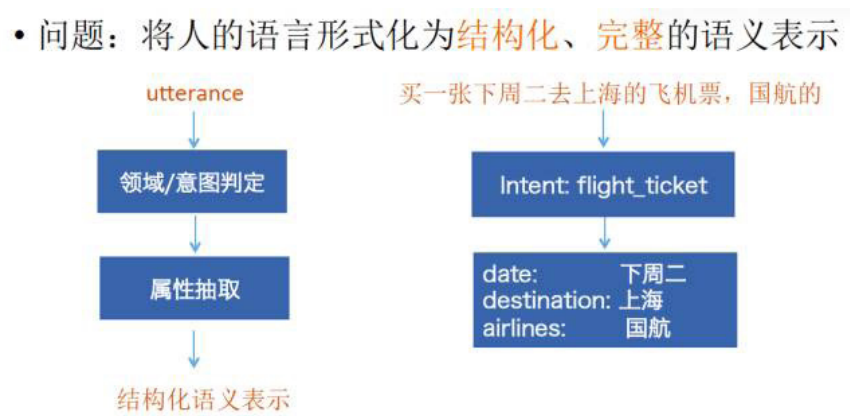
下面分享阿里巴巴在智能对话交互方向的进展和实践。先看对话交互逻辑的概况，传统的对话交互，大概会分以下几个模块：语音识别子系统会把语音自动转成文字；语言理解就是把用户说的文字转化成一种结构化的语义表示；对话管理就是根据刚才的语义理解的结果来决定采取什么样的动作 action，比如说定机票，或者设置闹钟。在语言生成这一块，就是根据 action 以及参数生成一段话，并通过一种比较自然的方式把它读出来。



对话系统架构简图

我认为现在人机交互和传统的人机交互一个不同点就是在数据和服务这一块，因为随着互联网的发展，数据和服务越来越丰富，那人机交互的目的是什么？归根到底还是想获取互联网的信息和各种各样的服务，所以，在现在的这种人机交互的场景中，数据和服务起的作用会越来越重要。

语言理解简单来说就是把用户说的话，转换为一种结构化的语义表示，从方法上会分成两个模块：用户意图的判定和属性的结构化抽取。来看一个例子，比如用户说，我要买一张下周去上海的飞机票，国航的。第一个模块要理解用户的意图是要买飞机票，第二，使用抽取模块，要把这些关键的信息出处理出来，出发时间、目的地、航空公司，从而得到一个比较完整的结构化的表示。



自然语言理解

人机对话中的语言理解面临哪些挑战呢？我总结为四类：

第一，表达的多样性。同样一个意图，比如用户说，我要听《大王叫我来巡山》，但是，不同的用户有太多太多不同的表达方式，我要听歌，给我放一首音乐等等。那对于机器来说，虽然表达方式不一样，但是意图是一样的，机器要能够理解这件事情。

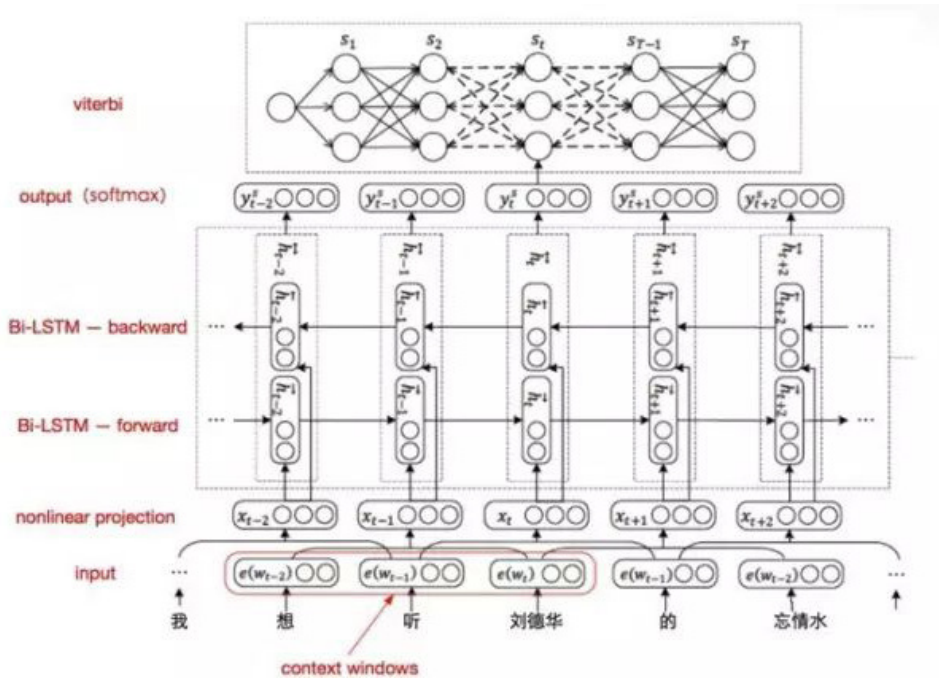
第二，语言的歧义性。比如说，我要去拉萨，它是一首歌的名字，挺好听的歌。当用户说，我要去拉萨的时候，他也可能是听歌，也可能是买一张去拉萨的机票，也可能是买火车票，或者旅游。 第三，语言理解的鲁棒性，因为用户说话过程当中，比较自然随意，语言理解要能够捕获住或者理解用户的意图。 第四，上下文的理解。这是人机对话交互与传统的 keyword 搜索一个非常大的不同，它的理解要基于上下文。

在语言理解这一块，我们把用户语言的意图理解抽象为一个分类问题，把它抽象为分类问题之后，就有一套相对标准的方法解决，比如 CNN 神经网络、SVM 分类器等等。阿里巴巴现在就是采用 CNN 神经网络方法，并在词的表示层面做了针对性的改进。机器要理解用户的话的意思，背后一定要依赖于大量的知识。比如说，“大王叫我来巡山”是一首歌的名字，“爱探险的朵拉”是一个视频，互联网上百万量级这样开放领域的实体知识并且每天都会有新的歌曲 / 视频出现，如果没有这样大量的知识，我觉得机器是很难真的理解用户的意图的。那么在词的语义表示这块，除了 word embedding，还引入了基于知识的语义表示向量。

刚才提到了，用户的话实际上是比较随意和自然的。那我们怎么样让这个模型有

比较好的鲁棒性来解决口语语言的随意性问题呢？我们针对用户标注的数据，通过算法自动加一些噪音，加了噪音之后（当然前提是不改变语义），然后基于这样的数据再 training 模型，模型会有比较好的鲁棒性。

第二个模块是属性抽取，在这一块，我们把它抽象为一个序列标注问题。这个问题，神经网络也有比较成型的方法，我们现在也是用这种双向 LSTM，在上面有一层 CRF 解码器，取得了不错的效果。当然其实在方法论上还是在神经网络框架下还是比较常见的或者说是标准的，但是这背后更大的功夫来自于对数据的分析和数据的加工。



以上所述的人机对话语言理解最大的特色就是基于上下文的理解，什么是上下文？我们看一个例子，用户说“北京天气怎么样?”，机器回答说“北京的天气今天温度 34 度”。接着用户说“上海”呢？在这里的理解一定是理解他指的是上海的天气，所以是要能够理解用户说的话，是和上文有关系的，所以我们再对问题做了一个抽象，在上下文的情况下，这句话和上文有关还是无关，把它抽象为二分的分类问题。这么做了抽象和简化以后，这个事情就有相对成型的方法。

刚才介绍的是语言理解。

对话引擎：就是根据语言理解的这种结构化的语义表示以及上下文，来决定采取什么样的动作。这个动作我们把它分成几类。第一，用于语言生成的动作。 第二，服务动作。 第三，指导客户端做操作的动作。

我们来看一个简单的对话例子。用户说我要去杭州，帮我订一张火车票，这个时候机器首先要理解用户的意图是买火车票，之后就要查知识库，要买火车票依赖于时间和目的地，但是现在用户只说目的地没说时间，所以它就要发起一个询问时间的动作，机器问了时间之后，用户回答说“明天上午”。这个时候机器要理解用户说的明天上午正好是在回答刚才用户问的问题，这样匹配了之后，基本上这个机器就把这个最关键的信息都收集回来了：时间和目的地。之后，机器就可以发起另外一个请求服务指令，然后把火车票的 list 给出来。这个时候用户接着说，“我要第二个”。机器还要理解用户说的第二个，就是指的要打开第二个链接，之后用户说“我要购买”，这个时候机器要发起一个指令去支付。

综上，对话交互，我会把它分成两个阶段：

第一阶段，通过多轮对话交互，把用户的需求表达完整，因为用户信息很多，不可能一次表达完整，所以要通过对话搜集完整，第一阶段得到结构化的信息（出发地、目的地、时间等）；有了这些信息之后，第二阶段就是请求服务，接着还要去做选择、确定、支付、购买等等后面一系列的步骤。

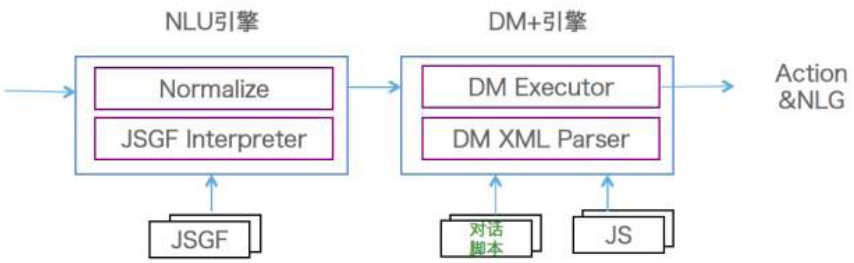
其实，传统的人机对话，包括现在市面上常见的人机对话，一般都是只在做第一阶段的对话，包括像大家熟知的亚马逊 Alexa，也只是在解决第一阶段的对话，第二阶段的对话做得不多。

我们在这个对话交互这块，在这方面还是做了一些有特色的东西。

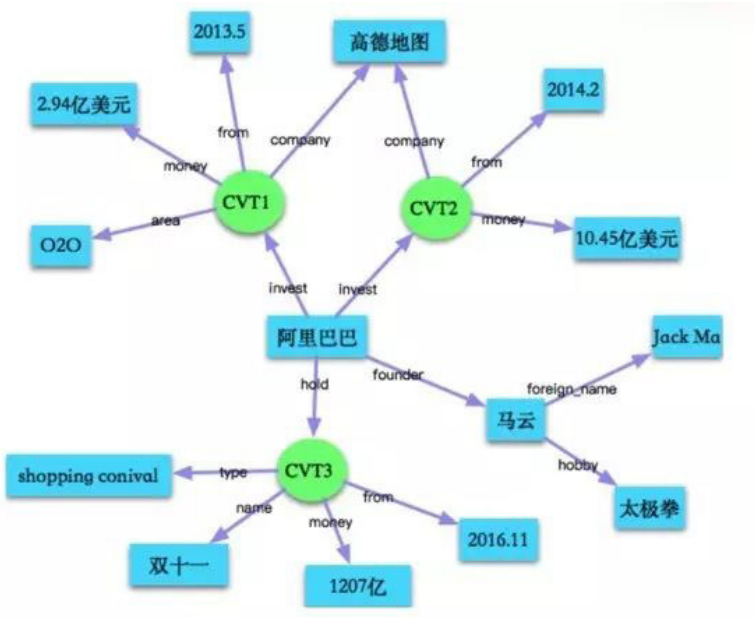
第一，我们设计了一套面向 Task Flow 的对话描述语言。刚才说了，对话其实是分两个阶段的。传统的对话只是解决了第一阶段，我们设计的语言能够把整个对话任务流完整地表达出来，这个任务流就是类似于咱们程序设计的流程图。对话描述语言带来的好处是它能够让对话引擎和业务逻辑实现分离，分离之后业务方可以开发脚本语言，不需要修改背后的引擎。

第二，由于有了 Task Flow 的机制，我们在对话引擎方带来的收益是能够实现对话的中断和返回机制。在人机对话当中有两类中断，一类是用户主动选择到另外一个意图，更多是由于机器没有理解用户话的意思，导致这个意图跳走了。由于我们维护了对话完整的任务流，知道当前这个对话处在一个什么状态，是在中间状态还是成功结束了，如果在中间状态，我们有机会让它回来，刚才讲过的话不需要从头讲，可以接着对话。

第三，我们设计了对话面向开发者的方案，称之为 Open Dialog，背后有一个语言理解引擎和一个对话引擎。面向开发者的语言理解引擎是基于规则办法，能够比较好的解决冷启动的问题，开发者只需要写语言理解的 Grammar、基于对话描述语言开发一个对话过程，并且还有对数据的处理操作。这样，一个基本的人机对话就可以完成了。



问答引擎：其实人和机器对话过程中，不仅仅是有 task 的对话，还有问答和聊天，我们在问答引擎这块，目前还是着力于基于知识图谱的问答，因为基于知识图谱的问答能够比较精准地回答用户的问题，所以我们在这方面下的力气会多一点。



聊天引擎：在聊天这块，我们设计了两类聊天引擎。

第一是基于 <K,V> 对的聊天引擎，它能够实现让业务方定制，为什么要定制呢？举个例子，在不同的场景下，机器的名字可能是不一样的，用户 A 和用户 B 给机器的名字是不一样的，我们有了这个机制之后，可以去定制机器的名字。但是这一方法的覆盖率是比较受限的。为了解决覆盖率的问题，我们又设计了基于 seq2seq 的生成式聊天引擎，这种生成式聊天，能够对开放的用户说的问题给出一个相对通顺并且符合逻辑的回答。当然这两类聊天引擎会有一个协同的策略在里面。

对话交互平台的开发策略以及赋能合作伙伴

刚才语言理解引擎、对话引擎、聊天引擎再加上语音识别合成，形成了完整的一套系统平台，我们称之为自然交互平台。在这套平台上，一端是连接着各种各样的设备，另外一端是连接了各种各样的服务，这样的话，用户和设备的交互就能够用比较自然的方式进行下去了。

值得一提的是，这样的自然交互平台在阿里巴巴已经有比较多的应用了。无论是在汽车、电视、音箱、机器人中，比如说在互联网汽车对话交互，我们和合作伙伴设计开发了汽车前装和汽车后装场景的对话交互。前装指的是我这个汽车在出厂的时候就有了对话交互能力，后装是指买了车之后再买一个设备，比如说行车记录仪、导航仪之类的。

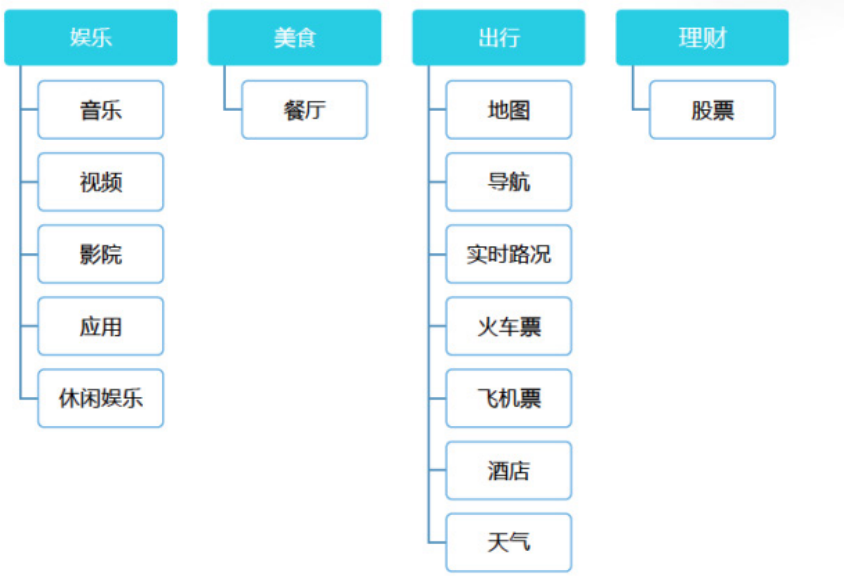
在功能上，比如说像地图、导航、路况，还有围绕着娱乐类的音乐、有声读物，还有实现对车的控制，在车的控制这块当然是受限的，现在能实现对车窗玻璃的控制。 在汽车场景下的对话交互，还和其他场景有非常多的不同。因为产品方希望当这个车在郊区网络不好的时候，最需要导航的时候，你要能够工作，所以我们的语音识别还有语言理解、对话引擎，就是在没有网络的情况下，要在端上能够完全工作，这里面的挑战也非常大。

有了这样一个对话交互平台，我们正在把这样的平台开放出来，让合作伙伴去开发自己场景的对话交互，所以我们正在开发面向开发者的平台，这个平台背后有端上的解决方案和云上的解决方案，端上包括声音的采集、VAD、端上无网情况下完整的对话方案，服务端的能力更强大一些。

在合作伙伴这块有两类。一类是面向设备的，比如说汽车、电视、音箱、机器人、智能玩具。 另外一类就是类似于行业应用，比如说智能客服这样的一个场景。

考察一个对话交互平台的能力，主要看它背后的沉淀和积累的核心能力，我们在这方面花了三年的时间去沉淀了一些公共场景的对话交互能力。比如像娱乐、出行、

理财、美食，有了这样的能力之后，当一个新的业务方接入的时候，就不需要再去开发了，直接调用就好。他只需要开发业务场景中特定的一些场景就可以，能够大大地加快业务方开发对话交互的速度。



其实对话交互平台背后第二个能力，就是提供足够强的定制能力，这种能力我们在语言理解，用户可以定制自己的时点、对话逻辑、聊天引擎、问答引擎，可以把自己积累的数据上传上来，以及对语音识别的词语定制，包括 TTS 声音的定制等等。

智能对话交互生态的范式思考

过去 3-4 年，在人机对话领域，应该是说还是取得了长足的进步，这样的进步来自于以深度学习为为代表的算法突破。这个算法的突破带来语音识别大的改进。同时，另一方面我们认为对话交互和真正的用户期望还是有明显的距离的，我认为有两点：第一，对话交互能覆盖的领域还是比较受限的，大家如果是用智能语音交互的产品，你就发现翻来覆去就是那几类，音乐、地图、导航、讲笑话等等。第二，有的能力体现得还不够好，所以我们想未来怎么办呢?

我们先看看现在的模式，有几种模式，我把它总结为两类模式。第一类，自主

研发。很多的创业公司或者是团队基本上都是自主研发的，像苹果它基本上是自主研发的模式。第二类，平台模式，比如说典型代表就是亚马逊的 Alexa，这个平台的一个好处，它能够发动开发者的力量快速地去扩展领域。但是你会发现现在在亚马逊 Alexa 上有 8000 多个这种能力，但是很多领域的体验都不够好，根本原因在于开放平台上开发的这些 skill 没有很好的评测手段和质量控制机制。

自主研发的好处是体验相对较好，平台模式的好处是能够快速扩展。所以如何把这两者结合在一起，有没有第三种模式。

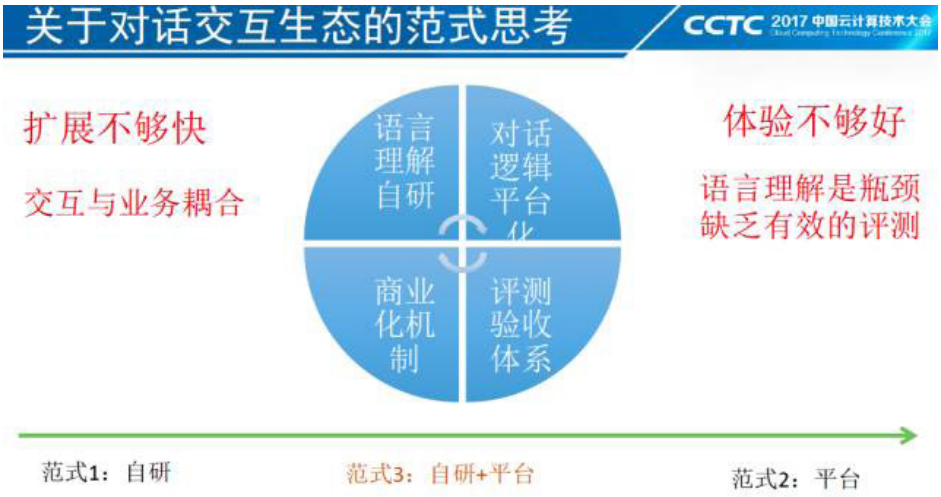
如果有，第三种模式应该有以下几个特点。

第一，由于自然语言理解的门槛还是比较高的，门槛高指的是对于开发者来说，它比开发一个 APP 难多了，从无到有开发出来不难，但要做到效果好是非常难的。所以，语言理解引擎最好能够自研；

第二，对话逻辑要平台化。对于对话交互因为它和业务比较紧，每个业务方有自己特殊的逻辑，通过平台化比较合适，让平台上的开发者针对各自场景的需求和交互过程来开发对话；

第三，需要建立一套评测体系，只有符合这个评测体系的，才能引入平台当中；

第四，需要商业化的机制，能够让开发者有动力去开发更多的以及体验更好的交互能力。



如果这几点能够做到，我们称之为第三种范式，基于这个范式的平台能够相对快速地并且开发的能力体验是有效果保证的。这样它开放给用户的时候，无论是对 B 用户还是 C 用户，可以有更多的用户价值。

总结

最后，总结下我们对于研发对话交互机器人的几点思考和体会：

第一，坚持用户体验为先。这个话说起来很容易，但是我们也知道，很多团队不是以用户为先的，是以投资者为先的，投资者喜欢什么样的东西，他们就开发什么样的东西。坚持用户体验为先，就是产品要为用户提供核心价值。

第二，降低产品和交互设计的不确定性。刚才也提到了，对话交互实际上最大的问题是不确定性，在产品的交互上我们要想办法把这种不确定性尽量降得低一点，惟有通过设备设计的时候降低，或者是交互界面的降低，或者是通过对话引导的方式，把这种不确定性尽量降低。

第三，打造语言理解的鲁棒性和领域扩展性。语言的理解能力尽量做到鲁棒性，能够比较好的可扩展。

第四，打造让机器持续学习能力。对话交互我认为非常非常重要的一点，就是怎么样能够让机器持续不断地学习。现在的这种对话交互基本上还都是有产品经理或者是人工定的，定好之后这个能力是固化的，所以怎么样能够把算法（策划学习）能力打造出来，就像小朋友学语言一样，随着年龄的增长，能力会持续不断地增强。

第五是打造数据闭环。要能够快速地达到数字闭环，当然这个闭环当中要把数据的效能充分调动起来，结合更多数据的服务。

本文根据干诀（博士，阿里巴巴资深专家）2017 年 5 月 18 在中国云计算技术大会上的演讲整理。

阿里史上首款 AI 硬件设备，为何如此“听话”？

阿里技术

7 月 6 日，阿里人工智能实验室发布了旗下首款智能语音终端设备天猫精灵 X1。天猫精灵 X1 内置第一代中文人机交流系统 AliGenie。AliGenie 生活在云端，它能够听懂中文普通话语音指令，目前可实现智能家居控制、语音购物、手机充值、叫外卖、音频音乐播放等功能，带来崭新的人机交互新体验。

天猫精灵 X1 和 AliGenie 均由阿里巴巴的科学家和工程师团队研发，应用了阿里巴巴积累多年的语音识别、自然语言处理、人机交互等技术。

一篇自然语言处理（NLP）的相关论文《一种新的语义编码模型及其在智能问答及分类中的应用》被国际数据挖掘顶会 KDD2017 收录，本次也被应用在天猫精灵里。在自然语言处理的两个核心应用场景——文本分类和智能问答上，天猫精灵这套「即刻唤醒，即刻识别」神经网络模型的智能问答准确率相比微软的 wikiqa 数据集和 IBM 的 insuranceqa 数据集提升了 2-4%，是目前业内最高水准。

本次阿里妹邀请到阿里巴巴人工智能实验室资深算法工程师王成龙，为大家深入解读这篇论文，揭开天猫精灵“听懂人话”的秘密。

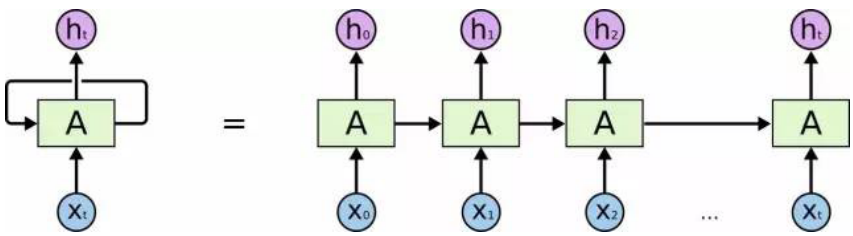
语义编码的意义

自然语言这一被人类发明的信号系统，通常被我们归为一种「非结构化数据」。其原因在于，自然语言文本是由一堆符号（token）顺序拼接而成的不定长序列，很难直接转变为计算机所能理解的数值型数据，因而无法直接进行进一步的计算处理。语义编码的目标即在于如何对这种符号序列进行数值化编码，以便于进一步地提取和应用其中所蕴含的丰富信息。语义编码是所有自然语言处理（Natural Language Processing，NLP）工作的「第一步」，同时也很大程度地决定了后续应用的效果。

传统的文本编码方式通常将其当作离散型数据，即将每个单词（符号）作为一个独立的离散型数值，如 Bag-of-Words (BOW)、TF-IDF 等。但是这类方法忽略了单词与单词之间的语义关联性，同时也难以对单词的顺序及上下文依赖信息进行有效编码。近几年，深度学习技术被广泛的应用于 NLP 领域，并在众多算法命题上取得了突破。其本质在于，深度神经网络在特征提取（语义编码）上具有极大的优势。

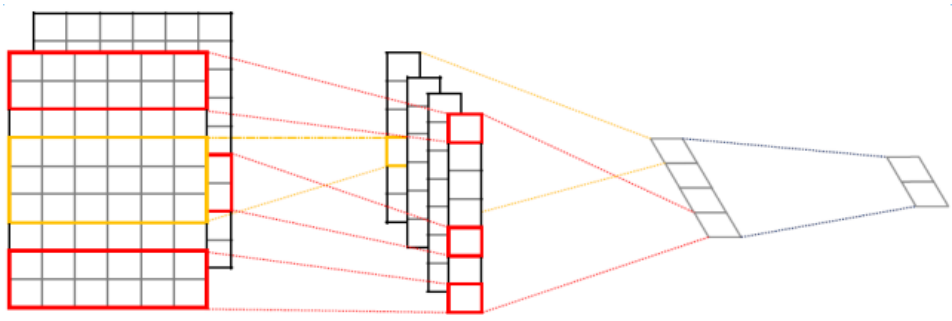
已有方法的瓶颈

当前，较为常用的文本语义编码模型包括循环神经网络（Recurrent Neural Network，RNN）以及卷积神经网络（Convolution Neural Network，CNN）。



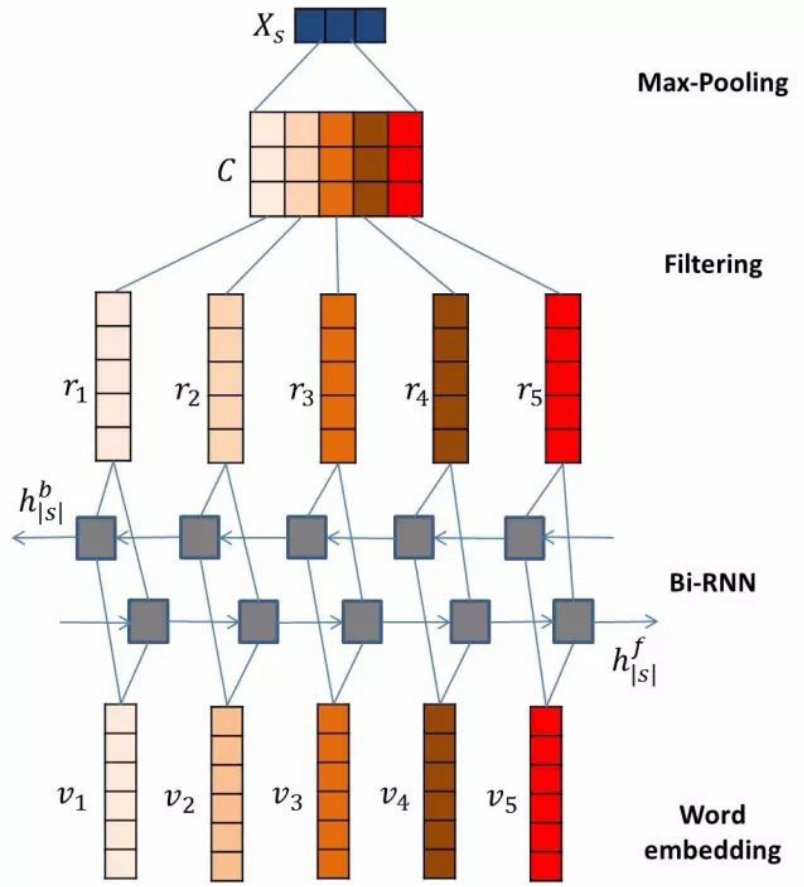
循环神经网络是应用最为广泛的序列数据神经网络建模方法。相对于传统的前向神经网络，循环神经网络的主要特点在于，在每个时刻，其输出不仅要依赖于当前时刻的输入，还要考虑上一时刻的模型「状态」。通过对历史状态的依赖，RNN 模型能

够有效的表征文本数据的上下文依存信息。但是，RNN 的「宏伟目标」– 有效管理任意跨度的信息传递 – 往往使得其难以有效的训练，进而也限制了其在具体应用中的效果。



另一被广泛应用的语义编码模型是 CNN 模型。传统的 CNN 建模通常用于解决图像的特征提取。但近年来，众多学者尝试将其应用到文本处理领域。CNN 的模型结构来源于对人类视觉神经信号处理机制的模拟。与文本数据不同的是，图像数据通常被看做一个二维数据结构，而相应的 CNN 模型也更适于提取其中的「局部」特征。但与图像数据相似的是，文本数据中的上下文依赖通常可以被简化为一种「局部」信息，即传统 NLP 领域中的 N-gram 语言模型：文本中一个词的具体含义，通常只和上文有限距离内的几个词相关。因此，CNN 中的「局部卷积」信息处理机制同样可以应用于文本数据中，用于提取文本中的 N-gram 特征。但是，与图像信息不同的是，文本数据中的上下文依赖关系有可能会经历一个很长的跨度。而 CNN 只能对固定范围内的局部依存关系进行建模。因此，CNN 语义编码方法也存在一定的缺陷。

Conv-RNN

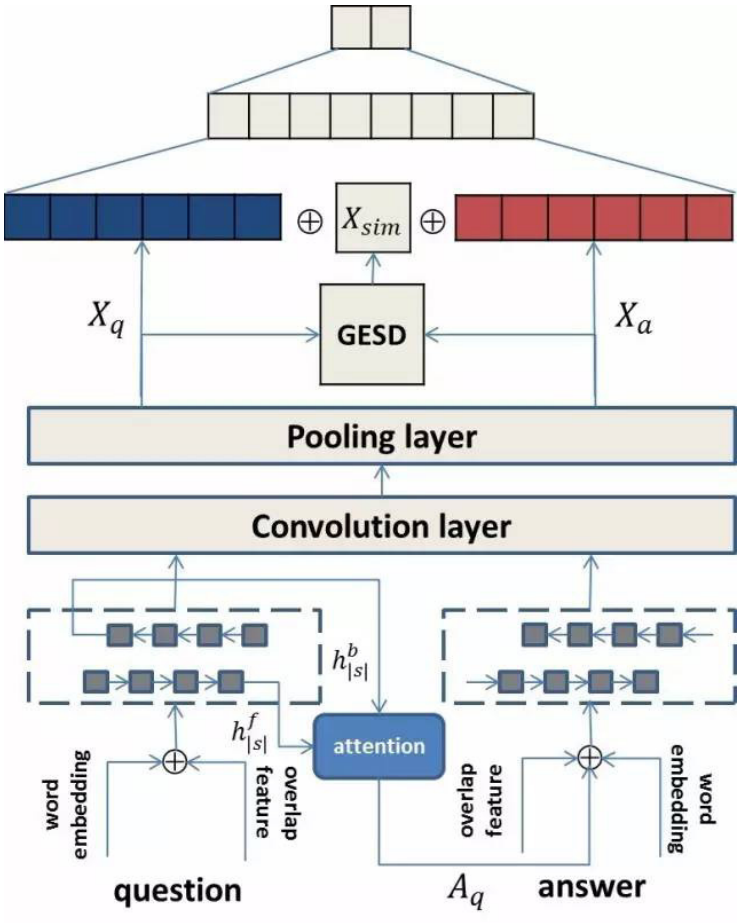


近期，我们团队与数据技术及产品部兄弟团队共同投稿一篇 KDD 文章，其中我们提出了一种新的文本语义编码算法 conv-RNN（如图 2 所示）。该模型在参考了循环神经网络与卷积神经网络的同时，进行了进一步的文本语义编码优化。conv-RNN 不仅保留了 RNN 模型对不定长跨度的上下文依赖的编码能力，还利用了 CNN 模型中常用的最大池化机制，用以更加简洁地从文本数据所蕴含的丰富信息中抽离出不同的信息表征。

此外，在 conv-RNN 的基础上，我们还提出了一种新的智能问答（answer selection）模型以及文本分类（sentence classification）模型。为了充分验证所提

出的模型的效果，我们分别选取了智能问答及文本分类领域的一批标准数据集，与当前业界的最新成果进行了对比验证。

智能问答



智能问答是当前比较火的一个 NLP 应用领域，也被认为是 NLP 研究最有可能于近期实现商业化落地的一个领域。在 conv-RNN 语义编码算法基础之上，我们进一步提出了一种新的问答匹配模型。此外，在该模型中，我们还引入了一种「权值共享」机制以及 attention 方法，用以进一步提升 question-answer 匹配效果。

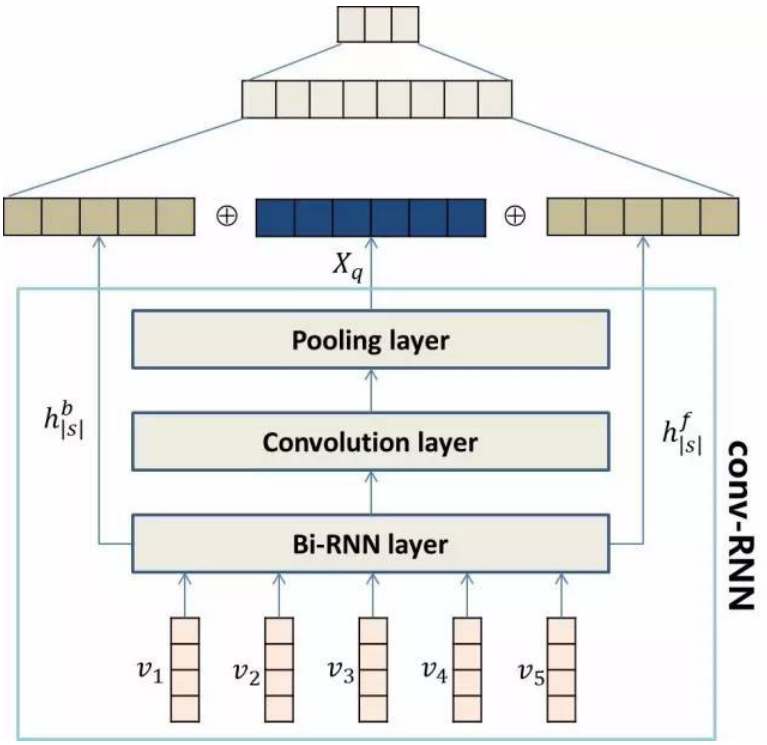
Model	MAP	MRR
Word Cnt [51]	0.4891	0.4924
Wgt Word Cnt [51]	0.5099	0.5132
LCLR [51]	0.5993	0.6086
Key-Value Memory Network [23]	0.7069	0.7265
DocChat+(2) [49]	0.7008	0.7222
Paragraph-Vec [51]	0.5110	0.5160
CNN [51]	0.6190	0.6281
Paragraph-Vec-Cnt [51]	0.5976	0.6058
CNN-Cnt [51]	0.6520	0.6652
CubeCNN [6]	0.7090	0.7234
NASM + Cnt [20]	0.689	0.707
L.D.C [48]	0.7058	0.7226
CNNr [35]	0.6951	0.7107
IARNN-Occam(context) [43]	0.7341	0.7418
PairwiseRank+SentLevel [33]	0.701	0.718
AP-CNN [34]	0.6886	0.6957
ABCNN [52]	0.6914	0.7127
$conv-RNN$	**0.7427**	**0.7504**

Model	dev	test1	test2
IR model [17]	52.7	55.1	50.8
QA-LSTM with attention [41]	68.4	68.1	62.2
CNN with GESD [4]	65.4	65.3	61.0
Attentive LSTM [40]	68.9	69.0	64.8
IARNN-Occam [43]	69.1	68.9	65.1
IARNN-Gate [43]	70.0	70.1	62.8
AP-BILSTM [34]	68.4	**71.7**	66.4
$conv-RNN$	**71.7**	71.4	**68.3**

我们选用了微软发布的 WikiQA 数据集以及 IBM 发布的 InsuranceQA 数据集用来对比所提出的模型与业界的 state-of-the-art 方法，以验证该模型的有效性。

由结果可知，在 WikiQA 数据集上，conv-RNN 击败了所有 state-of-the-art 方法，并且在 MAP（mean average precision）和 MRR（mean reciprocal rank）两个指标上均取得了较大的提升。在 InsuranceQA 数据集上，conv-RNN 在 dev 和 test2 两个测试集上均取得了较大的提升，仅在 test1 上略低于 AP-BILSTM。

文本分类



在 conv-RNN 的基础上，我们进一步提出了一种新的文本分类模型（如图 4 所示）。为了验证该模型的有效性，我们选取了业界常用的 5 个标准的分类数据集：Movie Review（MR）；Stanford Sentiment Treebank-1（SST-1）；Stanford Sentiment Treebank-2（SST-2）；Subj；IMDB。由对比结果可知，conv-RNN 在前 4 个数据集上均超越了各类 state-of-the-art 方法。

Model	MR	SST-1	SST-2	Subj	IMDB
Sent-Parser [3]	79.5	-	-	-	-
NBSVM [46]	79.4	-	-	93.2	91.32
MNB [46]	79.0	-	-	93.6	86.59
G-Dropout[47]	79.0	-	-	93.4	91.2
F-Dropout [47]	79.1	-	-	93.6	91.1
Drop-Bi [42]	-	-	-	-	91.98
NB-SVM Trigram [19]	-	-	-	-	91.87
Paragraph-Vec [14]	-	48.7	87.7	-	92.58
RAE [37]	77.7	43.2	82.4	-	-
MV-RNN [36]	79.0	44.4	82.9	-	-
RNTN [38]	-	45.7	85.4	-	-
DCNN [26]	-	48.5	86.8	-	-
CNN-non-static [10]	81.5	48.0	87.2	93.4	-
CNN-multichannel [10]	81.1	47.4	88.1	93.2	-
SA-LSTM [2]	-	-	-	-	92.76
WkA + 25% flexible [13] filters (FF)	80.02	46.11	84.29	92.68	90.16
Fully Connected [27] Layer Combination	81.59	-	-	-	-
Tree-CRF [28]	77.3	-	-	-	-
Tree-LSTM [39]	-	50.6	86.9	-	-
Multi-Task [16]	-	49.6	87.9	94.1	91.3
CCAE [7]	77.8	-	-	-	-
$conv - RNN$	81.99	51.67	88.91	94.13	90.39

总结

语义编码技术是所有 NLP 工作的基础，也是当前 NLP 技术进一步发展的主要「瓶颈」所在。我们在语义理解以及更上层的智能问答、多轮人机交互方向已经有了一定的技术积累，后续还会继续在这一方向发力，以期能够尽快做出为大众服务的人工智能产品。

史上最全！阿里智能人机交互的核心技术解析

孙健　李永彬　陈海青　邱明辉

阿里妹导读：过去 20 多年，互联网及移动互联网将人类带到了一个全新的时代，如果用一个词来总结和概括这个时代的话，“连接”这个词再合适不过了。这个时代主要建立了四种连接：第一，人和商品的连接；第二，人和人的连接；第三，人和信息的连接；第四，人和设备的连接。

“连接“本身不是目的，它只是为“交互”建立了通道。在人机交互（Human-Computer Interaction）中，人通过输入设备给机器输入相关信号，这些信号包括语音、文本、图像、触控等中的一种模态或多种模态，机器通过输出或显示设备给人提供相关反馈信号。“连接”为“交互”双方架起了桥梁。

“交互”的演进方向是更加自然、高效、友好和智能。对人来说，采用自然语言与机器进行智能对话交互是最自然的交互方式之一，但这条路上充满了各种挑战。如何让机器理解人类复杂的自然语言？如何对用户的提问给出精准的答案而不是一堆候选？如何更加友好地与用户闲聊而不是答非所问？如何管理复杂的多轮对话状态和对话上下文？在阿里巴巴，我们从 2014 年初开始对智能对话交互进行探索和实践创

新，研发成果逐步大规模应用在了智能客服（针对阿里巴巴生态内部企业的阿里小蜜、针对阿里零售平台上的千万商家的店小蜜，以及针对阿里之外企业及政府的云小蜜等）和各种设备（如 YunOS 手机、天猫魔盒、互联网汽车等）上。

本文将对阿里巴巴在智能对话交互技术上的实践和创新进行系统的介绍。首先简要介绍智能对话交互框架和主要任务；接下来详细介绍自然语言理解、智能问答、智能聊天和对话管理等核心技术；然后介绍阿里巴巴的智能对话交互产品；最后是总结和思考。强烈建议收藏细看！

1. 智能对话交互框架

典型的智能对话交互框架如图 1 所示。其中，语音识别模块和文本转语音模块为可选模块，比如在某些场景下用户用文本输入，系统也用文本回复。自然语言理解和对话管理是其中的核心模块，广义的自然语言理解模块包括对任务类、问答类和闲聊类用户输入的理解，但在深度学习兴起后，大量端到端（End-to-End）的方法涌现出来，问答和聊天的很多模型都是端到端训练和部署的，所以本文中的自然语言理解狭义的单指任务类用户输入的语义理解。在图 2 所示的智能对话交互核心功能模块中，自然语言理解和对话管理之外，智能问答用来完成问答类任务，智能聊天用来完成闲聊类任务。在对外输出层，我们提供了 SaaS 平台、PaaS 平台和 BotFramework 三种方式，其中 Bot Framework 为用户提供了定制智能助理的平台。

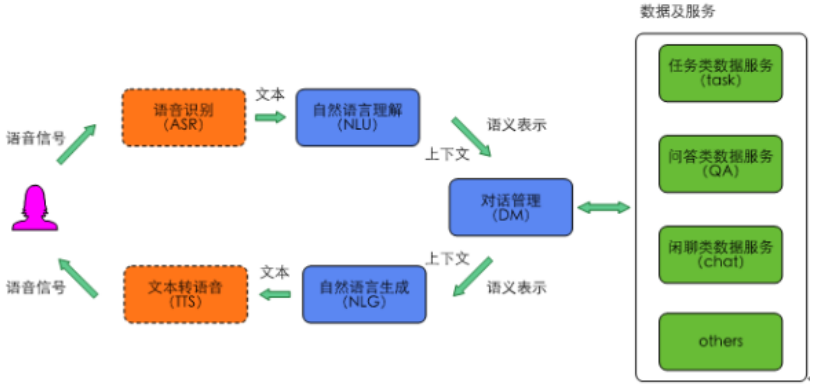


图 1　智能对话交互框架

2. 智能对话交互核心技术

智能对话交互中的核心功能模块如图 2 所示，本部分详细介绍智能对话交互中除输出层外的自然语言理解、智能问答、智能聊天和对话管理四个核心模块。

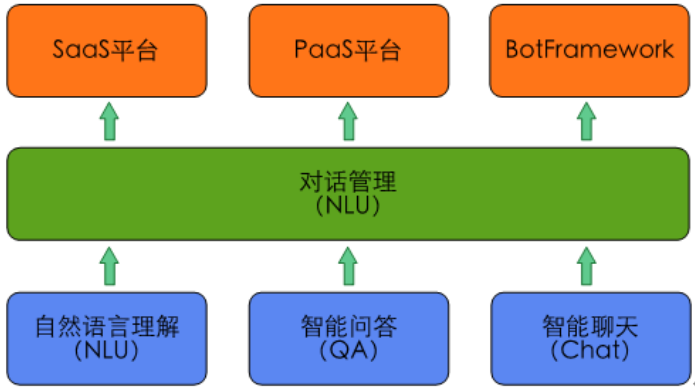


图 2　智能对话交互中的核心功能模块

2.1　自然语言理解

自然语言理解是人工智能的 AI-Hard 问题 [1]，也是目前智能对话交互的核心难题。机器要理解自然语言，主要面临如下的 5 个挑战。

（1）语言的多样性　　（2）语言的多义性　　（3）语言的表达错误

（4）语言的知识依赖　　（5）语言的上下文

表1　上下文示例

U：上海明天的天气 A：上海明天天气…… U：后天呢	U：那你嫁给我吧 A：我妈说我还小 U：我问过你妈了她说同意你嫁给我
继续延续问天气	如何正确的把闲聊接下去

注：U 指用户（user），A 指智能体（agent）。下同。

整个自然语言理解围绕着如何解决以上难点问题展开。

2.1.1 自然语言理解语义表示

自然语言理解的语义表示主要有三种方式[2]。

（1）分布语义表示（Distributional semantics）

（2）框架语义表示（Frame semantics）

（3）模型论语义表示（Model-theoretic semantics）

在智能对话交互中，自然语言理解一般采用的是 framesemantics 表示的一种变形，即采用领域（domain）、意图（intent）和属性槽（slots）来表示语义结果，如图 3 所示。

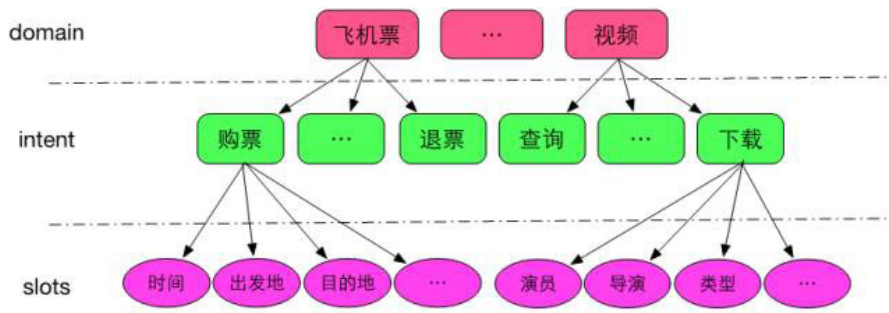


图 3 domain ongology 示意图

在定义了上述的 domain ontology 结构后，整个算法流程如图 4 所示。

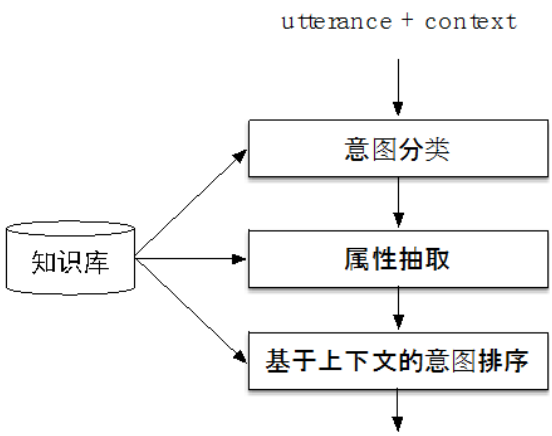


图 4 自然语言理解流程简图

2.1.2 意图分类

意图分类是一种文本分类，主要分为基于规则的方法、基于传统机器学习的方法和基于深度学习的方法，如 CNN [3]、LSTM [4]、RCNN [5]、C-LSTM [6] 及 FastText[7] 等。针对 CNN、LSTM、RCNN、C-LSTM 四种典型的模型框架，我们在 14 个领域的数据集上进行训练，在 4 万左右规模的测试集上进行测试，采用 Micro F1 作为度量指标（注：此处的训练和测试中，神经网络的输入只包含 word embedding，没有融合符号表示），结果如图 5 所示，其中 Yoon Kim 在 2014 年提出的基于 CNN[3] 的分类算法效果最好。

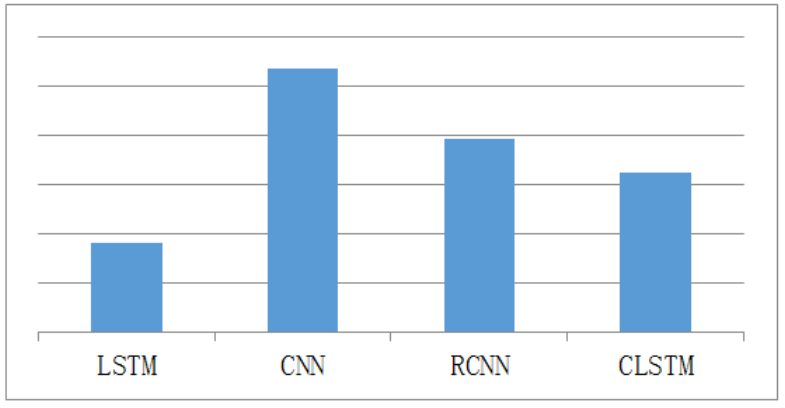


图 5 四种模型的分类效果对比

单纯以 word vector 为输入的 CNN 分类效果，在某些领域上无法超越复杂特征工程的 SVM 分类器。如何进一步提升深度学习的效果，其中一个探索方向就是试图把分布式表示和符号表示进行融合。比如对于“刘德华的忘情水”这句话，通过知识库可以标注刘德华为 singer、忘情水为 song，期望能把 singer 和 song 这样的符号表示融入到网络中去。具体融合方法，既可以把符号标签进行 embedding，然后把 embedding 后的 vector 拼接到 wordvector 后进行分类，也可以直接用 multi-hot 的方式拼接到 word vector 后面。分布式表示和符号表示融合后的 CNN 结构如图 6 所示。

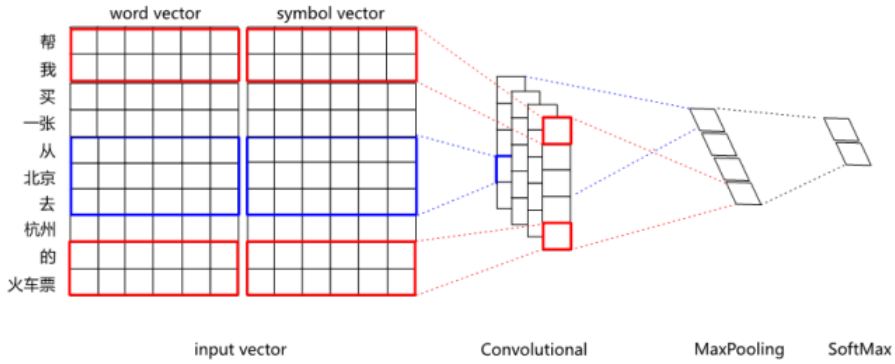


图 6　分布式表示和符号表示融合后的 CNN 分类网络结构

经过融合后，在 14 个领域约 4 万条测试数据集上，对比融合前后的 F1 值（如图 7 所示），从中可以看出，像餐厅、酒店、音乐等命名实体多且命名形式自由的领域，效果提升非常明显。

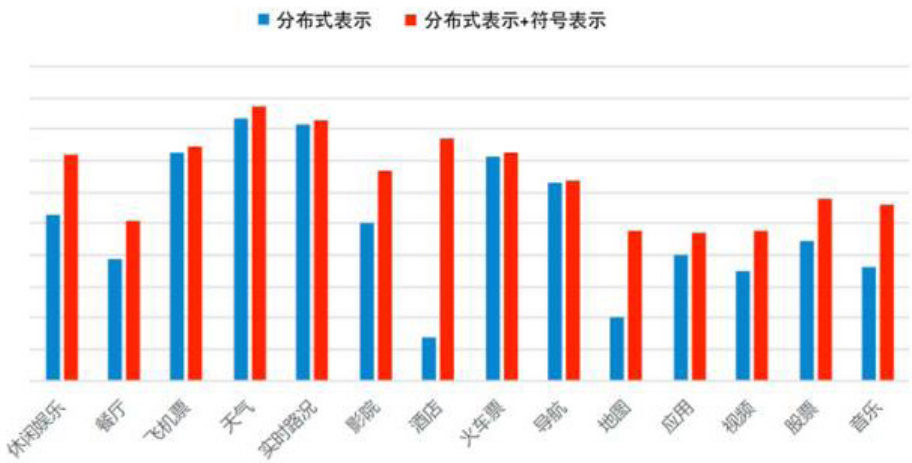


图 7　在 CNN 中分布式表示融合符号表示前后效果对比

在以词为输入单位的 CNN 中，经常会遇到 OOV（Out-Of-Vocabulary）问题，一般情况下会使用一个特殊向量（比如固定的随机向量或者已知词向量的平均值）来表示所有的 OOV，这样做的效果肯定不够好。在我们的实现中，引入了 FastText [8]

来训练 word vector，对于 OOV，可以用其 subword 向量计算得到，有效地解决了 OOV 的问题。

在效果优化方面，除了本文中所述的 word vector 的动态训练和 dropout 之外，通过对训练数据进行数据增强（data augmentation），效果会有较大的提升。

2.1.3 属性抽取

属性抽取问题可以抽象为一个序列标注问题，可以以字为单位进行序列标注，也可以以词为单位进行序列标注，如图 8 所示为以词为单位进行序列标注的示例。在这个例子中包含 departure、destination 和 time 三个待标注标签；B 表示一个待标注标签的起始词；I 表示一个待标注标签的非起始词，O 表示非待标注标签词。

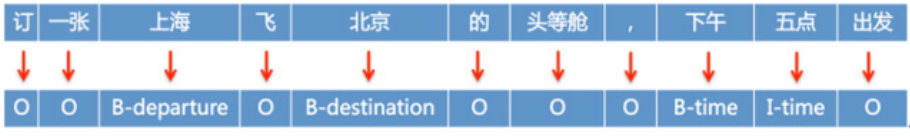


图 8 序列标注示例

属性抽取的方法，包括基于规则的方法，基于传统统计模型的方法，经典的如 CRF[9]，以及基于深度学习模型的方法。2014 年，在 ARTIS 数据集上，RNN[10] 模型的效果超过了 CRF。此后，R-CRF [11]、LSTM[12]、Bi-RNN[13]、Bi-LSTM-CRF[14] 等各种模型陆续出来。

在属性抽取这个任务中，我们采用了如图 9 的网络结构，该结构具有以下优点。

（1）输入层

在输入层，我们做了三部分工作：① 采用了分布式表示（word vector）和符号表示（symbol vector）融合的方式，有效利用了分布式的上下文学习能力和符号的抽象知识表示能力；②采用了局部上下文窗口（local context window），将窗口内的词的表示拼接在一起送入一个非线性映射层，非线性映射具有特征学习和特征降维的作用；③采用了 FastText [8] 进行 word embedding 的学习，可以有效解决 OOV（Out-Of-Vocabulary）的问题。

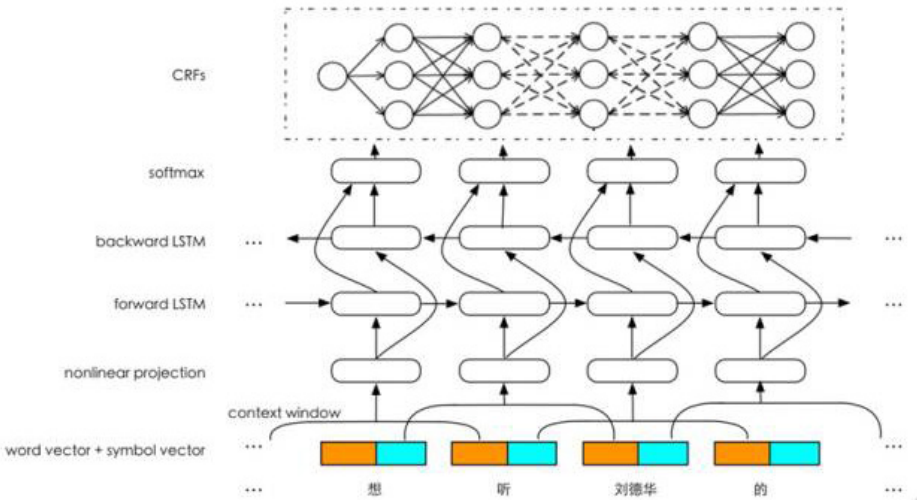


图 9　属性抽取网络结构

（2）Bi-LSTM 层

在中间的隐藏层，采用 Bi-LSTM 进行特征学习，既能捕捉上文特征，也能捕捉下文特征。

（3）输出层

在输出层有几种典型的做法，比如 Bi-LSTM+Softmax、Bi-LSTM+CRF 等，Bi-LSTM+Softmax 是把属性抽取在输出层当成了一个分类问题，得到的标注结果是局部最优，Bi-LSTM+CRF 在输出层会综合句子层面的信息得到全局最优结果。

2.1.4 意图排序

在表 1 中，我们展示了一个例子，如果不看上下文，无法确定“后天呢”的意图。为了解决这个问题，在系统中我们设计了意图排序模块，其流程如图 10 所示。对于用户输入的 utterance，一方面先利用分类抽取模型去判定意图并做抽取；另一方面，直接继承上文的意图，然后根据这个意图做属性抽取。这两个结果通过特征抽取后一起送入一个 LR 分类器，以判定当前 utterance 是应该继承上文的意图，还是遵循分类器分类的意图。如果是继承上文意图，那么可以把这个意图及其属性抽取结果作为最终结果输出；如果是遵循分类器分类的结果，那么可以把各个结果按照分类

器分类的置信度排序输出。

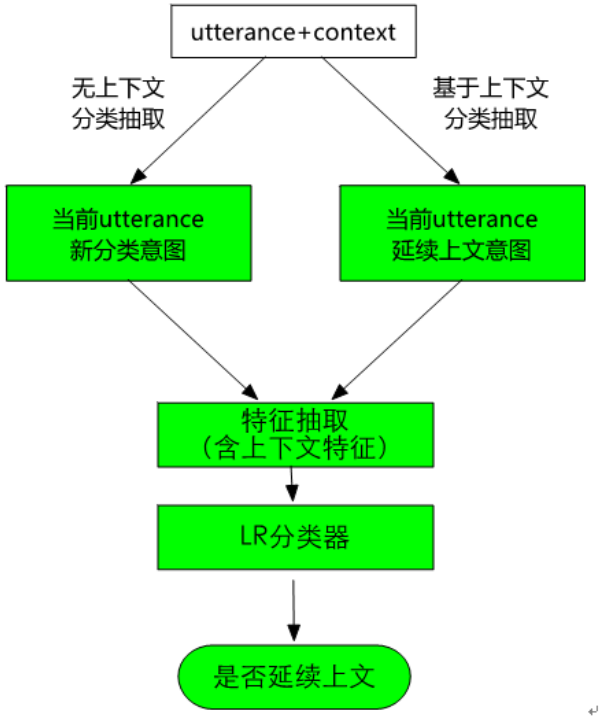


图 10　基于上下文的意图延续判定

2.2　智能问答

在具体的业务场景中有三种典型的问答任务，一是用户提供 QA-Pairs，一问一答；二是建立结构化的知识图谱，进行基于知识图谱的问答；三是针对非结构化的文本，进行基于阅读理解的问答。本文重点介绍我们在阅读理解方面做的工作，比如利用阅读理解解决淘宝活动规则的问答。

在阅读理解的方法上，目前针对斯坦福大学的数据集 SquAD，有大量优秀的方法不断涌现，比如 match-LSTM [15]、BiDAF [16]、DCN [17]、FastQA [18] 等。文献 [18] 给出了目前的通用框架，如图 11 所示，主要分为 4 层：① Word Embedder，对问题和文档中的词进行 embedding；② Encoder，对问题和文档进行编码，一般采用 RNN/LSTM/BiLSTM；③ Interaction Layer（交互层），在问题和文档之间

逐词进行交互，这是目前研究的热点，主流方法是采用注意力机制（attention）；④ Answer Layer（答案层），预测答案的起始位置和结束位置。

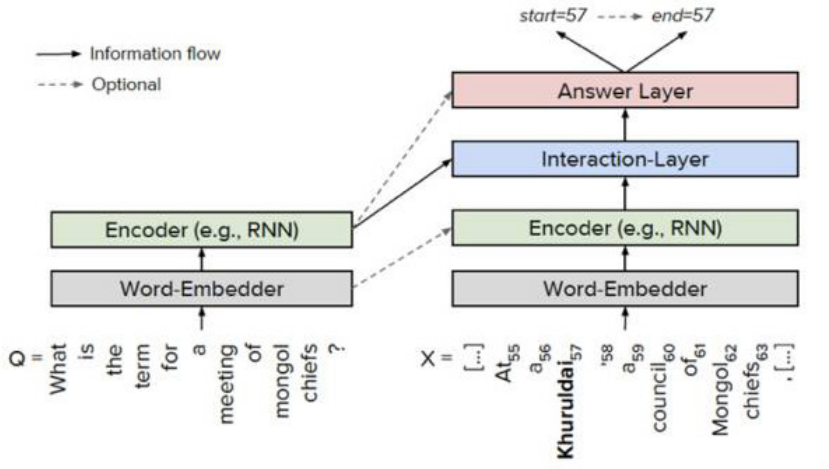


图 11　阅读理解的通用框架

我们在具体实现中，参考 BiDAF [16] 网络结构，在此基础上做了大量优化。

（1）模型的业务优化

需要改进模型的结构设计，使得模型可以支持电商文档格式的输入。电商规则文档往往包含大量的文档结构，如大小标题和文档的层级结构等，将这些特定的篇章结构信息一起编码输入到网络中，将大幅提升训练的效果。

（2）模型的简化

学术文献中的模型一般都较为复杂，而工业界场景中由于对性能的要求，无法将这些模型直接在线上使用，需要做一些针对性的简化，使得模型效果下降可控的情况下，尽可能提升线上预测性能，例如可以简化模型中的各种 bi-lstm 结构。

（3）多种模型的融合

当前这些模型都是纯粹的 end-to-end 模型，其预测的可控性和可解释性较低，要适用于业务场景的话，需要考虑将深度学习模型与传统模型进行融合，达到智能程度和可控性的最佳平衡点。

2.3 智能聊天

面向 open domain 的聊天机器人目前无论在学术界还是在工业界都是一大难题，目前有两种典型的方法：一是基于检索的模型，比如文献 [19-20]，其基本思路是利用搜索引擎通过计算相关性来给出答案；二是基于 Seq2Seq 的生成式模型，典型的方法如文献 [21-22]，其网络结构如图 12 所示。

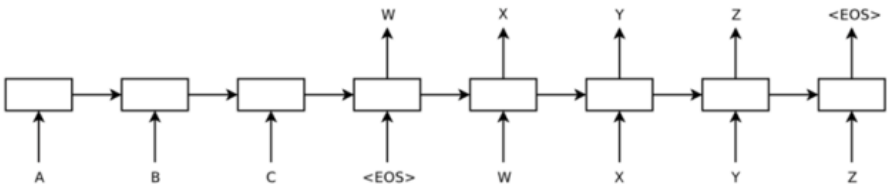


图 12　Seq2Seq 典型网络结构

检索模型的优点是答案在预设的语料库中，可控，匹配模型相对简单，可解释性强；缺点是在一定程度上缺乏对语义的理解，且有固定语料库的局限性，长尾问题覆盖率较差。生成模型的优点是通过深层语义方式进行答案生成，答案不受语料库规模限制；缺点是模型的可解释性不强，且难以保证回答一致性和合理性。

在我们的聊天引擎中，结合检索模型和生成模型各自的优势，提出了一种新的模型 AliMe Chat [23]，基本流程如图 13 所示。首先采用检索模型从 QA 知识库中找出候选答案集合；然后利用带注意力的 Seq2Seq 模型对候选答案进行排序，如果第一候选的得分超过某个阈值，则作为最终答案输出，否则利用生成模型生成答案。其中带注意力的 Seq2Seq 模型结构如图 14 所示。经过训练后，主要做了如下测试：如图 15 所示，利用 600 个问题的测试集，测试了检索（IR）、生成（Generation）、检索 + 重排序（Rerank）及检索 + 重排序 + 生成（IR+Rerank+Generation）四种方法的效果，可以看到在阈值为 0.19 时，IR+Rerank+Generation 的方法效果最好。

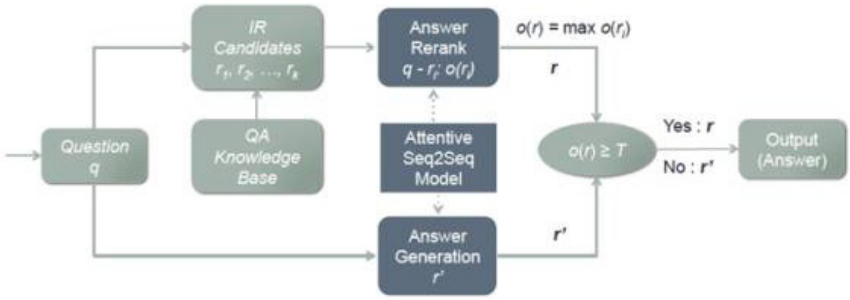


图 13　AliMe Chat 流程图

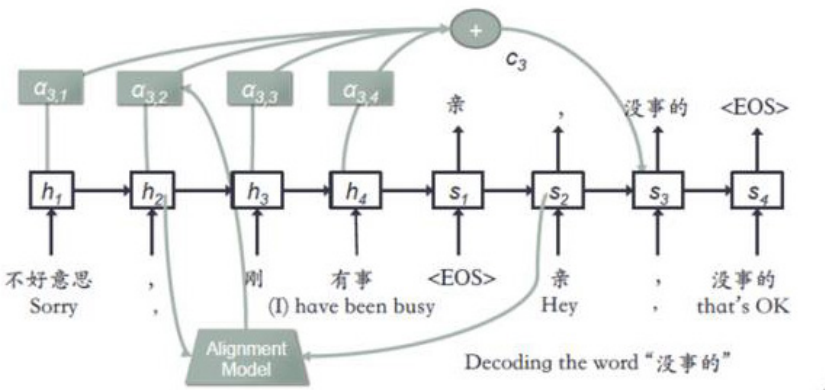


图 14　带注意力的 Seq2Seq 网络结构示例

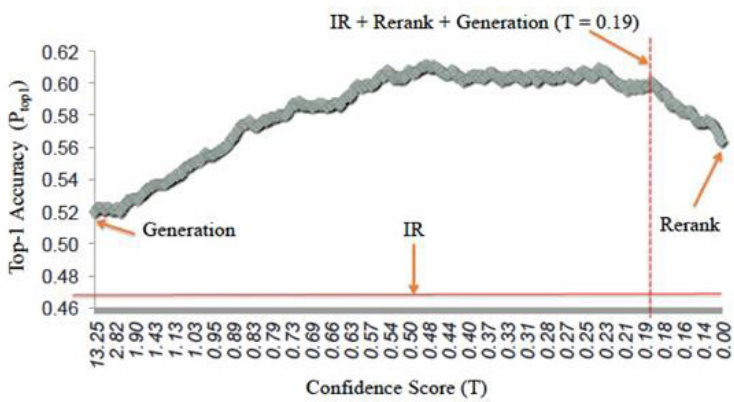


图 15　IR、Generation、Rerank、IR+Rerank+Generation 效果对比

此模型在阿里小蜜中上线后，示例如图 16 所示。在阿里小蜜中，针对之前的 IR 模型和 AliMe Chat 模型，利用线上流量做了 A/B Test，结果如图 17 所示。从用户日志中随机选择 2 136 条数据，其中 1 089 是采用 IR 模型回答，另外 1 047 是采用 AliMe Chat 回答，AliMe Chat Top1 答案的准确率（accuracy）是 60.36%，远远好于 IR 的 40.86%。

图 16　AliMe Chat 在阿里小蜜中上线后的聊天示例

Model	N_{total}	$N_{unsuitable}$	$N_{neutral}$	$N_{suitable}$	P_{top_1}
IR	1089	644	384	61	40.86%
Hybrid	1047	415	504	128	60.36%

图 17　阿里小蜜中 IR 方法与 AliMe Chat 方法 A/BTest 结果

2.4 对话管理

对话管理根据语言理解的结构化语义表示结果以及上下文，来管理整个对话的状态，并决定下一步采取什么样的动作。

下面来看一个简单的对话例子。

U：我要去杭州，帮我订一张火车票

A：请问你什么时间出发?

U：明天上午

A：为你找到了以下火车票：

U：我要第二个

A：第二个是……，您是否要购买?

U：我要购买

对话交互分成两个阶段，第一阶段，通过多轮对话交互，把用户的需求收集完整，得到结构化的信息（出发地、目的地、时间等）；第二阶段就是请求服务，接着还要去做选择、确定、支付、购买等后面一系列的步骤。

传统的人机对话，包括现在市面上常见的人机对话，一般都是只在做第一阶段的对话，第二阶段的对话做得不多。对此，我们设计了一套对话管理体系，如图 18 所示，这套对话管理体系具有以三个特点。

第一，设计了一套面向 Task Flow 的对话描述语言。该描述语言能够把整个对话任务流完整地表达出来，这个任务流就是类似于程序设计的流程图。对话描述语言带来的好处是它能够让对话引擎和业务逻辑实现分离，分离之后业务方可以开发脚本语言，不需要修改背后的引擎。

第二，由于有了 Task Flow 的机制，我们在对话引擎方带来的收益是能够实现对话的中断和返回机制。在人机对话当中有两类中断，一类是用户主动选择到另外一个意图，更多是由于机器没有理解用户话的意思，导致这个意图跳走了。由于我们维护了对话完整的任务流，知道当前这个对话处在一个什么状态，是在中间状态还是成

功结束了，如果在中间状态，我们有机会让它回来，刚才讲过的话不需要从头讲，可以接着对话。

第三，设计了对话面向开发者的方案，称之为 OpenDialog，背后有一个语言理解引擎和一个对话引擎。面向开发者的语言理解引擎是基于规则办法，能够比较好地解决冷启动的问题，开发者只需要写语言理解的 Grammar，基于对话描述语言开发一个对话过程，并且还有对数据的处理操作。这样，一个基本的人机对话就可以完成了。

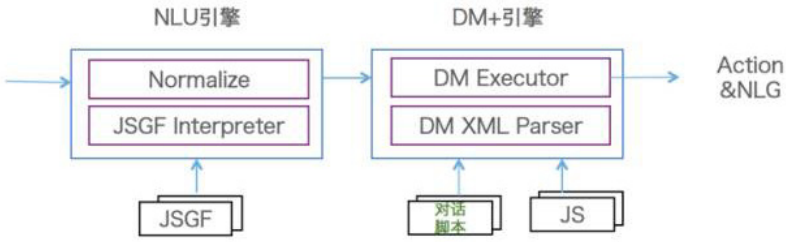


图 18　对话管理框架图

3. 阿里巴巴智能对话交互产品

3.1　智能服务——小蜜家族

2015 年 7 月，阿里巴巴推出了自己的智能服务助理 – 阿里小蜜，一个围绕着电子商务领域中的服务、导购，以及任务助理为核心的智能对话交互产品。通过电子商务领域与智能对话交互领域的结合，带来传统服务行业模式的变革与体验的提升。在 2016 年的双“十一”期间，阿里小蜜整体智能服务量达到 643 万，其中智能解决率达到 95%，智能服务在整个服务量 (总服务量 = 智能服务量 + 在线人工服务量 + 电话服务量) 占比也达到 95%，成为了双“十一”期间服务的绝对主力。阿里小蜜主要服务阿里国内业务和阿里国际化业务，国内业务如淘宝、天猫、飞猪、健康、闲鱼、菜鸟等，国际化业务如 Lazada、PayTM、AE 等。

随着阿里小蜜的成功，将智能服务能力赋能给阿里生态圈商家及阿里生态之外的企业和政府部门，便成了必然的路径。店小蜜主要赋能阿里生态中的商家，云小蜜则面向阿里之外的大中小企业、政府等。整个小蜜家族如图 19 所示。

图 19　小蜜家族

3.2　智能设备

过去 3~4 年，我们可以看到，连接互联网的设备发生了很大变化，设备已经从 PC 和智能手机延伸到更广泛的智能设备，比如智能音箱、智能电视、机器人、智能汽车等设备。智能设备的快速发展正在改变着人和设备之间的交互方式。

我们研发的智能对话交互平台为各种设备提供对话交互能力，目前在 YunOS 手机、天猫魔盒、互联网汽车等设备上已经大量应用。比如在天猫魔盒中，用户通过对话交互可以完成搜视频、查音乐、问天气等，可以进行闲聊，还可以进行购物。

4. 总结与思考

过去几年中，结合阿里巴巴在电商、客服、智能设备方面的刚性需求和场景，我们在智能对话交互上做了大量的探索和尝试，构建了一套相对完整的数据、算法、在线服务、离线数据闭环的技术体系，并在智能服务和智能设备上得到了大规模的应用，简单总结如下。

（1）自然语言理解方面，通过 CNN/Bi-LSTM-CRF 等深度学习模型、分布式表示和符号表示的融合、多粒度的 wordembedding、基于上下文的意图排序等方法，构建了规则和深度学习模型有机融合的自然语言理解系统。

（2）智能问答方面，成功的将机器阅读理解应用在了小蜜产品中。

（3）智能聊天方面，提出了 AliMe Chat 模型，融合了搜索模型和生成模型的优点，大大提高了闲聊的精度。

（4）对话管理方面，设计了基于 Task Flow 的对话描述语言，将业务逻辑和对话引擎分离，并能实现任务的中断返回和属性的 carry-over 等复杂功能。

在智能交互技术落地应用的过程，我们也在不断思考怎么进一步提高智能交互技术水平和用户体验。

第一，坚持用户体验为先。坚持用户体验为先，就是产品要为用户提供核心价值。

第二，提高语言理解的鲁棒性和领域扩展性。

第三，大力发展机器阅读理解能力。

第四，打造让机器持续学习能力。

第五，打造数据闭环，用数据驱动效果的持续提升。

目前的人工智能领域仍然处在弱人工智能阶段，特别是从感知到认知领域需要提升的空间还非常大。智能对话交互在专有领域已经可以与实际场景紧密结合并产生巨大价值，尤其在智能客服领域（如阿里巴巴的小蜜）。随着人工智能技术的不断发展，未来智能对话交互领域的发展还将会有不断的提升。

阿里巴巴集团 – 智能服务事业部招募自然语言理解、人机对话、知识图谱和智能问答等方向的人才啦！该职位将致力于自然语言的基础研究和开发（NLP）、自然语言的语义理解，打造全球领先的智能人机对话交互平台（阿里小蜜、店小蜜、云小蜜），服务于各行各业的企业 / 组织。 研发方向包括但不限于：

1. 自然语言理解算法和技术的研究；

2. 人机对话模型的算法和技术的研究；

3. 知识图谱和智能问答相关算法和技术的研究；

4. 针对人工智能客服机器人的应用和产品研发；

如果你愿意和我们一起创造人机交互的未来，热烈欢迎加入我们！

联系方式：jian.sun@alibaba-inc.com(千诀)

参考文献

[1] https://en.wikipedia.org/wiki/Natural_language_understanding.
[2] Percy Liang, Natural Language Understanding:Foundations and State-of-the-Art,ICML, 2015.
[3] Yoon Kim, Neural Networks for Sentence Classification, EMNLP, 2014.
[4] Suman Ravuri, and Andreas Stolcke, Recurrent Neural Network and LSTM Modelsfor Lexical Utterance Classification, InterSpecch, 2015.
[5] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao, Recurrent ConvolutionalNeural Networks for Text Classification, AAAI, 2015.
[6] Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis C.M. Lau, A C-LSTMNeural Network for Text Classification, arXiv, 2015.
[7] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov, Bag ofTricks for Efficient Text Classification, EACL, 2017.
[8] Piotr Bojanowski,Edouard Grave, Armand Joulin and Tomas Mikolov, Enriching-Word Vectors withSubword Information, TACL, 2017.
[9] C. Raymond, and G.Riccardi, Generative and discriminative algorithms for spoken languageunderstanding, Interspeech, 2007.
[10] Kaisheng Yao,Geoffrey Zweig, Mei-Yuh Hwang, Yangyang Shi, and Dong Yu, Recurrent neural networksfor language understanding, InterSpeech, 2013.
[11] Kaisheng Yao,Baolin Peng, Geoffrey Zweig, Dong Yu, Xiaolong Li, and Feng Gao, Recurrent conditionalrandom field for language understanding, ICASSP, 2014.
[12] Kaisheng Yao,Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi, Spokenlanguage understanding using long short-term memory neural networks, 2014 IEEESpoken Language Technology Workshop (SLT), 2014.
[13] Gr é goire Mesnil,Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, XiaodongHe, Larry Heck, Gokhan Tur, Dong Yu, and Geoffrey Zweig, Using Recurrent NeuralNetworks for Slot Filling in Spoken Language Understanding, TASLP, 2015
[14] Guillaume Lample,Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer,Neural architectures for named entity recognition, NAACL, 2016.
[15] Shuohang Wang, and Jing Jiang, Machine comprehension using match-lstm andanswer pointer, ICLR, 2017.
[16] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hananneh Hajishirzi,Bidirectional attention flow for machine comprehension, ICLR, 2017.
[17] Caiming Xiong, Victor Zhong, and Richard Socher, Dynamic coattention networks-for question answering, ICLR, 2017.
[18] Dirk Weissenborn, Georg Wiese, and Laura Seiffe, Making Neural QA as Simple asPossible but not Simpler, arXiv, 2017.
[19] Zongcheng Ji,Zhengdong Lu, and Hang Li, An information retrieval approach to short textconversation, arXiv, 2014.
[20] Zhao Yan, NanDuan, Jun-Wei Bao, Peng Chen, Ming Zhou, Zhoujun Li, and

Jianshe Zhou, Docchat:An information retrieval approach for chatbot engines using unstructureddocuments, ACL, 2016.
[21] Dzmitry Bahdanau,Kyunghyun Cho, and Yoshua Bengio, Neural machine translation by jointly learningto align and translate, ICLR, 2015.
[22] Oriol Vinyals, andQuoc V. Le., A neural conversational model, ICML Deep Learning Workshop, 2015.
[23] Minghui Qiu,Feng-Lin Li, Siyu Wang, et al., AliMe Chat: A Sequence to Sequence and Rerankbased Chatbot Engine, ACL, 2017.

深度学习要多深，才能读懂人话？| 阿里小蜜前沿探索

吉仁

阿里妹注：本篇文章全面阐述了“机器阅读理解综述及在电商领域的探索”主题，总字数近五千字，预计需要 10 分钟左右的阅读时间。推荐对深度学习、大数据、自然语言处理感兴趣的童鞋收藏。

文末有个温馨提醒，请相关童鞋留意哦。

研究背景

阿里小蜜是阿里巴巴推出的围绕电商服务、导购以及任务助理为核心的智能人机交互产品，在去年双十一期间，阿里小蜜整体智能服务量达到 643 万，其中智能解决率达到 95%，成为了双十一期间服务的绝对主力。阿里小蜜所使用的问答技术也在经历着飞速的发展，从最初基于检索的知识库问答演进到了最新的语义深度建模。

近期，阿里小蜜正在开展新的探索，让机器具有如同人一般的阅读理解能力，这将使得问答产品体现出真正的智能，进一步提升服务效率。

在问答技术中，最常见的是检索式问答，但存在很多的限制，例如，需要人工进行知识提炼，让机器在事先准备好的问答对基础之上进行检索。而我们经常需要对这样一段话进行提问：

5 月 18 日，阿里巴巴集团发布了 2017 财年 Q4 财报和 2017 财年全年的业绩报告。财报显示，阿里巴巴集团 2017 财年的收入为 1582.73 亿元人民币，同比增长 56%；经调整后净利为 578.71 亿元人民币。Q4 集团收入为 385.79 亿元，同比增长 60%；非美国通用会计准则下的净利同比增长 38% 至 104.4 亿元人民币。

Q：阿里巴巴 2017 财年收入是多少呢?

A：1582.73 亿人民币

其中包含很多的数字、日期、金额以及一些客观事实描述，如果对每个提问点都设置一个问答对，将会是非常繁琐的过程，同时，由于真实场景中问题的长尾性，人

工提炼也无法穷尽所有可能的问题，知识往往覆盖率比较低。如果能脱离人工提炼知识的过程，直接让机器在非结构化文本内容中进行阅读理解，并回答用户的问题，将是一个里程碑式的进步。

深度学习近年来在自然语言处理领域中广泛应用，在机器阅读理解领域也是如此，深度学习技术的引入使得机器阅读理解能力在最近一年内有了大幅提高，因此我们尝试结合深度学习与真实业务场景，探索机器阅读在电商领域的创新与落地场景。

电商领域的应用场景

那么机器阅读理解及问答技术在电商领域会有哪些合适的应用场景呢?

阿里小蜜的交易规则解读类场景

首先是电商交易规则解读类场景，例如在每次双 11 等活动时，都会有大量的用户对活动规则进行咨询。以往，阿里小蜜的知识运营同学都需要提前研究淘宝和天猫上的活动规则，从一堆规则描述、活动介绍文本中提炼可能的问题。而通过机器阅读理解的运用，则可以让机器直接对规则进行阅读，为用户提供规则解读服务，是最自然的交互方式。

2016 年淘宝双十一消费者交易规则

发货时间说明：

2016 年 11 月 11 日 00:00:00-2016 年 11 月 17 日 23:59:59 期间付款成果的订单，确认收到时间如下：

（1）如果选择的物流方式为快递与货运，自“卖家已发货”状态起的 15 天后，系统会自动确认收货。

（2）如果选择物流方式为平邮，自“卖家已发货”状态起的 30 天后，系统会自动确认收货。

示例提问：我双十一买的东西什么时候会自动确认收货?

店小蜜的商品售前咨询场景

店小蜜是一款面向淘系千万商家的智能客服。用户在淘宝和天猫上购物时，往往会对较为关注的商品信息进行询问确认后才会下单购买，例如“荣耀 5c 的双摄像头拍照效果有什么特点?”而这些信息往往已经存在于商品的详情描述页，通过机器阅读理解技术，可以让机器对详情页中的商品描述文本进行更为智能地阅读和回答，降低服务成本的同时提高购买转化率。

相关工作调研

基于知识库自动构建的机器阅读

用传统的自然语言处理方式完成基于机器阅读理解的问答，一般需要先在文本中进行实体和属性的解析，构建出结构化的知识图谱，并在知识图谱基础上进行问答。主要涉及以下几个过程：

- 实体检测：识别文本中提及的实体，对实体进行分类。
- 实体链接：将检测出来的实体与知识库中现有实体进行匹配和链接。
- 属性填充：从文档中检测实体的各类预先定义好的属性，补充到知识库中。
- 知识检索：在知识库中根据实体和属性找到最相关的答案作为回答。

显然，用传统的知识库构建方式来进行机器阅读，虽然其可控性和可解释性较好，但领域垂直特点较强，难以适应多变的领域场景，且技术上需要分别解决多个传统 NLP 中的难点，如命名实体识别、指代消解、新词发现、同义词归一等，而每个环节都可能引入误差，使得整体误差逐渐扩大。

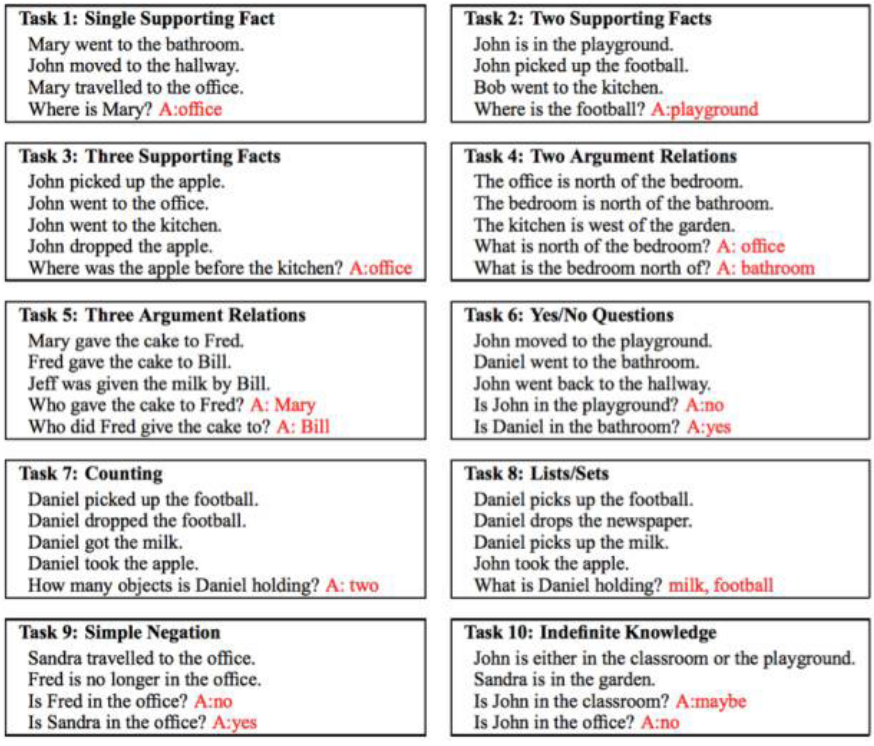
基于 End-to-end 的机器阅读

近年来，一些高质量的公开数据集为机器阅读领域的研究带来了革命性的变化，推动了基于 End-to-end 方法的高速进步，基于不同的数据集可以解决一些特定的机器阅读理解任务。以下首先介绍每一类中具有代表性的数据集。

- Facebook 的 bAbI 推理型问答数据集 [1]

bAbI 数据集是由人工构造的由若干简单事实形成的英文文章，给出文章和对应问题后，要求机器阅读理解文章内容并作出一定的推理，从而得出正确答案，往往是文章中的某个（或几个）关键词或者实体。

数据集包含对 20 个具体任务的评测，包括 Supporting Fact、Yes/No Question、Counting、Time Reasoning、Position Reasoning、Path Finding 等等。数据集规模相对比较小，仅由 1000 个训练数据和 1000 个测试数据构成，每个任务 150 个词。



• Microsoft 的 MCTest 选择题数据集 [2]

MCTest 数据集用于回答选择题，由真实的英文儿童读物构成。给出一篇 150–300 词的故事文章，并对故事内容进行提问，要求机器在几个候选答案中选出正确的答案。数据集的数据量较小，分为 160 篇和 500 篇两种。

James the Turtle was always getting in trouble. Sometimes he'd reach into the freezer and empty out all the food. Other times he'd sled on the deck and get a splinter. His aunt Jane tried as hard as she could to keep him out of trouble, but he was sneaky and got into lots of trouble behind her back.
One day, James thought he would go into town and see what kind of trouble he could get into. He went to the grocery store and pulled all the pudding off the shelves and ate two jars. Then he walked to the fast food restaurant and ordered 15 bags of fries. He didn't pay, and instead headed home.
His aunt was waiting for him in his room. She told James that she loved him, but he would have to start acting like a well-behaved turtle.
After about a month, and after getting into lots of trouble, James finally made up his mind to be a better turtle.

1) What is the name of the trouble making turtle?
A) Fries
B) Pudding
C) James
D) Jane

2) What did James pull off of the shelves in the grocery store?
A) pudding
B) fries
C) food
D) splinters

3) Where did James go after he went to the grocery store?
A) his deck
B) his freezer
C) a fast food restaurant
D) his room

4) What did James do after he ordered the fries?
A) went to the grocery store
B) went home without paying
C) ate them
D) made up his mind to be a better turtle

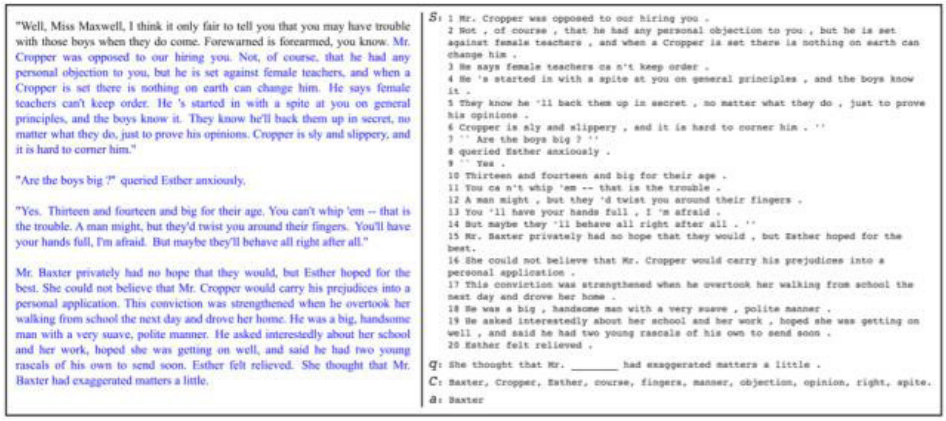
• DeepMind 的 CNN\DailyMail 完形填空数据集 [3]

该数据集来自于真实的新闻数据，但由自动标注而得，并非人工标注。给出一篇新闻和一个问句，把问句中的某个实体抽掉，要求能正确预测被抽掉的实体。这个实体必须是在文中出现过的。该数据集的数据量相对比较大，CNN9 有万篇，DailyMail 有 22 万篇。

Original Version	Anonymised Version
Context	
The BBC producer allegedly struck by Jeremy Clarkson will not press charges against the "Top Gear" host, his lawyer said Friday. Clarkson, who hosted one of the most-watched television shows in the world, was dropped by the BBC Wednesday after an internal investigation by the British broadcaster found he had subjected producer Oisin Tymon "to an unprovoked physical and verbal attack." ...	the *ent381* producer allegedly struck by *ent212* will not press charges against the " *ent153* " host , his lawyer said friday . *ent212* , who hosted one of the most - watched television shows in the world , was dropped by the *ent381* wednesday after an internal investigation by the *ent180* broadcaster found he had subjected producer *ent193* " to an unprovoked physical and verbal attack . " ...
Query	
Producer **X** will not press charges against Jeremy Clarkson, his lawyer says.	producer **X** will not press charges against *ent212* , his lawyer says .
Answer	
Oisin Tymon	*ent193*

• Facebook 的 CBT 完形填空数据集 [4]

该数据集也是由真实的儿童读物，由自动标注方式构成，其形式是给出 21 个句子，前 20 个是完整的供机器阅读的句子，将第 21 个句子中的实体抽掉，要求能正确预测出来。



"Well, Miss Maxwell, I think it only fair to tell you that you may have trouble with those boys when they do come. Forewarned is forearmed, you know. Mr. Cropper was opposed to our hiring you. Not, of course, that he had any personal objection to you, but he is set against female teachers, and when a Cropper is set there is nothing on earth can change him. He says female teachers can't keep order. He 's started in with a spite at you on general principles, and the boys know it. They know he'll back them up in secret, no matter what they do, just to prove his opinions. Cropper is sly and slippery, and it is hard to corner him."

"Are the boys big ?" queried Esther anxiously.

"Yes. Thirteen and fourteen and big for their age. You can't whip 'em -- that is the trouble. A man might, but they'd twist you around their fingers. You'll have your hands full, I'm afraid. But maybe they'll behave all right after all."

Mr. Baxter privately had no hope that they would, but Esther hoped for the best. She could not believe that Mr. Cropper would carry his prejudices into a personal application. This conviction was strengthened when he overtook her walking from school the next day and drove her home. He was a big, handsome man with a very suave, polite manner. He asked interestedly about her school and her work, hoped she was getting on well, and said he had two young rascals of his own to send soon. Esther felt relieved. She thought that Mr. Baxter had exaggerated matters a little.

S: 1 Mr. Cropper was opposed to our hiring you .
2 Not , of course , that he had any personal objection to you , but he is set against female teachers , and when a Cropper is set there is nothing on earth can change him .
3 He says female teachers ca n't keep order .
4 He 's started in with a spite at you on general principles , and the boys know it .
5 They know he 'll back them up in secret , no matter what they do , just to prove his opinions .
6 Cropper is sly and slippery , and it is hard to corner him . ''
7 '' Are the boys big ? ''
8 queried Esther anxiously .
9 '' Yes .
10 Thirteen and fourteen and big for their age .
11 You ca n't whip 'em -- that is the trouble .
12 A man might , but they 'd twist you around their fingers .
13 You 'll have your hands full , I 'm afraid .
14 But maybe they 'll behave all right after all . ''
15 Mr. Baxter privately had no hope that they would , but Esther hoped for the best .
16 She could not believe that Mr. Cropper would carry his prejudices into a personal application .
17 This conviction was strengthened when he overtook her walking from school the next day and drove her home .
18 He was a big , handsome man with a very suave , polite manner .
19 He asked interestedly about her school and her work , hoped she was getting on well , and said he had two young rascals of his own to send soon .
20 Esther felt relieved .

Q: She thought that Mr. _____ had exaggerated matters a little .

C: Baxter, Cropper, Esther, course, fingers, manner, objection, opinion, right, spite.

a: Baxter

• 讯飞和哈工大的中文完形填空数据集 [5]

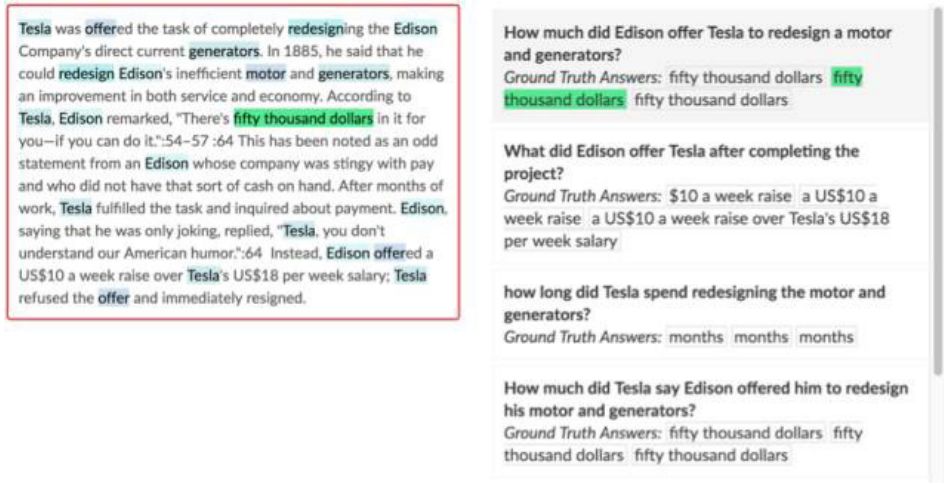
这是一份中文数据集，和 CBT 类似，这份数据集是根据真实的新闻数据由自动标注的方式获得的完形填空数据集，数据集较大，共有 87 万篇。

```
1 ||| 人民网 1月 1日 讯 据 《 纽约 时报 》 报道 ， 美国 华尔街 股市 在 2013年 的 最后 一 天 继续 上涨 ，
 和 全球 股市 一样 ， 都 以 最高 纪录 或 接近 最高 纪录 结束 本年 的 交易 。
2 ||| 《 纽约 时报 》 报道 说 ， 标普 500 指数 今年 上升 29.6% ， 为 1997年 以来 的 最 大 涨幅 ；
3 ||| 道琼斯 工业 平均 指数 上升 26.5% ， 为 1996年 以来 的 最 大 涨幅 ；
4 ||| 纳斯达克 上涨 38.3% 。
5 ||| 就 12月 31日 来说 ， 由于 就业 前景 看好 和 经济 增长 明年 可能 加速 ， 消费者 信心 上升 。
6 ||| 工商 协进会 报告 ， 12月 消费者 信心 上升 到 78.1 ， 明显 高于 11月 的 72 。
7 ||| 另 据 《 华尔街 日报 》 报道 ， 2013年 是 1995年 以来 美国 股市 表现 最 好 的 一 年 。
8 ||| 这 一 年 里 ， 投资 美国 股市 的 明智 做法 是 追 着 “ 傻钱 ” 跑 。
9 ||| 所谓 的 “ 傻钱 ” X ， 其实 就 是 买 入 并 持有 美国 股票 这样 的 普通 组合 。
10 ||| 这个 策略 要 比 对冲 基金 和 其它 专业 投资者 使用 的 更为 复杂 的 投资 方法 效果 好 得 多 。
11 ||| 所谓 的 “ 傻钱 ” X ， 其实 就 是 买 入 并 持有 美国 股票 这样 的 普通 组合 。 ||| 策略
```

• Stanford 的 SQuAD 可变长答案数据集 [6]

SQuAD 是斯坦福发布的英文可变长答案数据集。问题和答案由大量众包人力来标注。内容主要以 Wiki 文章为主，涵盖了体育、政治、商业等各种领域的内容。数据集包含 500 多篇文章（在 1000 篇文章中随机选取，保证数据分布广泛），再将 500 多篇文章拆成了 2 万多个小段落，再对段落进行提问，共 10 万个问题（每个段落提 5 个以上的问题）。

每个问题需要由 3 位标注者进行标注，其中第一位标注者需要提问和回答，后两位标注者只需回答第一位标注者的提问。取第二位标注者的答案作为 Human Predict，取第一、三位标注者的答案作为 Ground Truth，以此获得多个 Ground Truth 来降低标注误差。且鼓励标注者用自己的语言来提问，不要去抄原文中的话。



该数据集一经推出便引发学术界的持续关注，在 SQuAD 的 Leadeboard 上不断有新的模型提出一遍遍刷新 benchmark，截止目前（2017 年 6 月），最好的方法已经获得超过 84 的 F1 值。值得一提的是，Standford 在推出该数据集时，基于传统 Lexical 特征工程的方式构造了 Baseline（考虑 Word/POS/Dependency..），作为白盒模型有较好的比较意义，同时也提供了人工评测的 Acc 作参照，方便衡量问题上界。

Rank	Model	EM	F1
1 Mar 2017	r-net (ensemble) *Microsoft Research Asia* http://aka.ms/rnet	76.922	84.006
2 May 2017	MEMEN (ensemble) *Eigen Technology & Zhejiang University*	75.37	82.658
3 Mar 2017	ReasoNet (ensemble) *MSR Redmond*	75.034	82.552
4 May 2017	r-net (single model) *Microsoft Research Asia* http://aka.ms/rnet	74.614	82.458
5 May 2017	Mnemonic Reader (ensemble) *NUDT & Fudan University* https://arxiv.org/abs/1705.02798	73.754	81.863

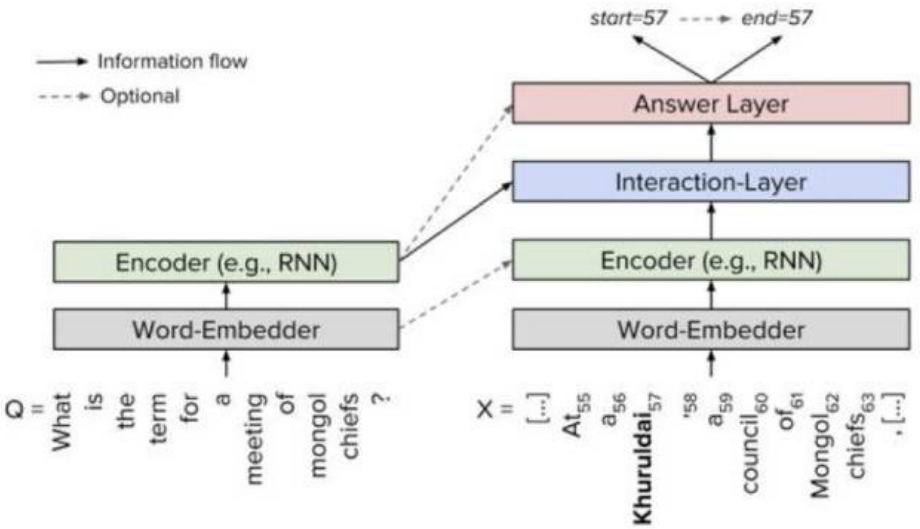
几乎所有围绕 SQuAD 的模型都可以概括为类似的结构：

（1）Embedding 层：将原文和问题中的词汇映射为向量表示。

（2）Encoding 层：用 RNN 对原文和问题进行编码，使得每个词蕴含上下文语义信息。

（3）Interaction 层：用于捕捉问题和原文之间的交互关系，并输出编码了问题语义信息的 query-aware 的原文表示。

（4）Answer 层：基于 query-aware 的原文表示来预测答案开始和结束的位置。



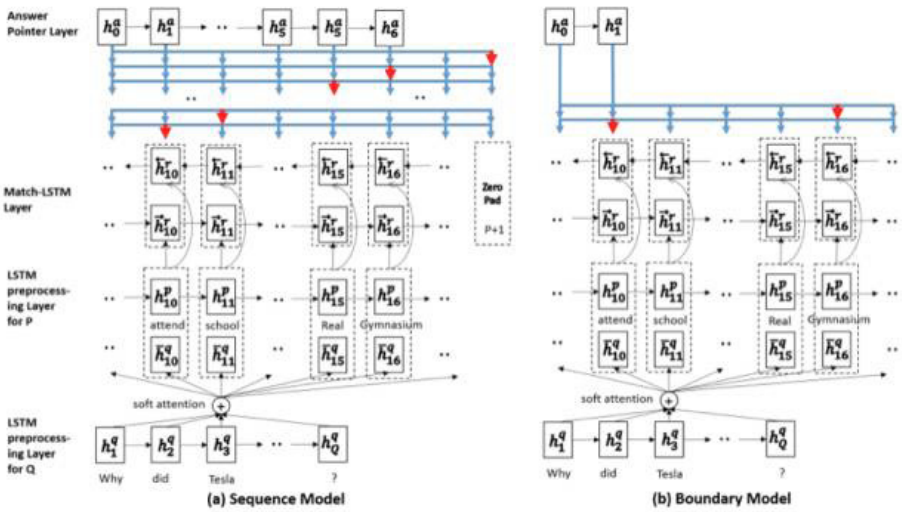
显然，SQuAD 数据集相比之前的完形填空、选择题型数据集来说，数据量更大、数据质量更高、解决的问题也更复杂，同时也更接近于真实的业务场景，因为在大部分的真实问答场景中，答案都并非单个实体，很可能是一个短句，因此我们将主要围绕 SQuAD LeaderBoard 上榜的模型来作一些具体的介绍。

基于 SQuAD 的机器阅读模型

• Match-LSTM with AnswerPointer [7]

Match-LSTMwith Answer Pointer 模型是较早登上 SQuAD LeaderBoard 的模型，作者提出了融合 match-lstm 和 pointer-network 的机器阅读框架，后续也被多篇相关工作所借鉴。其中的 Boundary Model 由于只预测答案开始和结束位置，

极大地缩小了搜索答案的空间，使得整个预测过程得到了简化。



模型主要包括三个部分：

（1）用 LSTM 分别对 question 和 passage 进行 encoding；

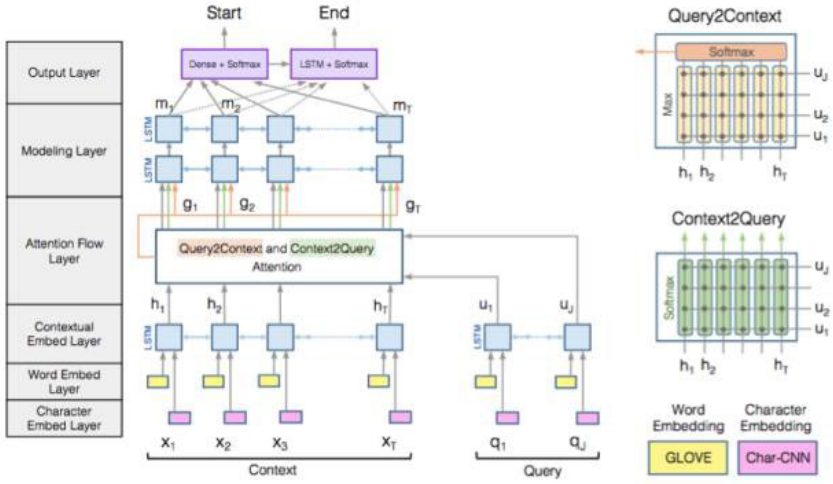
（2）用 match-LSTM 对 question 和 passage 进行 match。这里将 question 当做 premise，将

passage 当做 hypothesis，用标准的 word-by-word attention 得到 attention 向量，然后对 question 的隐层输出进行加权，并将其跟 passage 的隐层输出进行拼接，得到一个新的向量，然后输入到 LSTM；

（3）利用 pointer net 从 passage 中选择 tokens 作为答案。包括 Sequence Model 和 Boundary Model。其中 Sequence Model 是在 passage 中选择不连续的 word 作为 answer；BoundaryModel 只需要 passage 的起始位置和中止位置得到连续的 words 作为 answer。

- Bidirectional AttentionFlow [8]

BiDAF 模型最大的特点是在 interaction 层引入了双向注意力机制，计算 Query2Context 和 Context2Query 两种注意力，并基于注意力计算 query-aware 的原文表示。



模型由这样几个层次组成：

（1）Character Embedding Layer 使用 char-CNN 将 word 映射到固定维度的向量空间；

（2）Word Embedding Layer 使用（pre-trained）word embedding 将 word 映射到固定维度的向量空间；

（3）Contextual Embedding Layer 将上面的到的两个 word vector 拼接，然后输入 LSTM 中进行 context embedding；

（4）Attention Flow Layer 将 passage embedding 和 question embedding 结合，使用 Context-to-query Attention 和 Query-to-contextAttention 得到 word-by-word attention;

（5）Modeling Layer 将上一层的输出作为 bi-directional RNN 的输入，得到 Modeling 结果 M；

（6）Output Layer 使用 M 分类得到 passage 的起始位置，然后使用 M 输入 bi-directional LSTM 得到 M2，再使用 M2 分类得到 passage 的中止位置作为 answer。

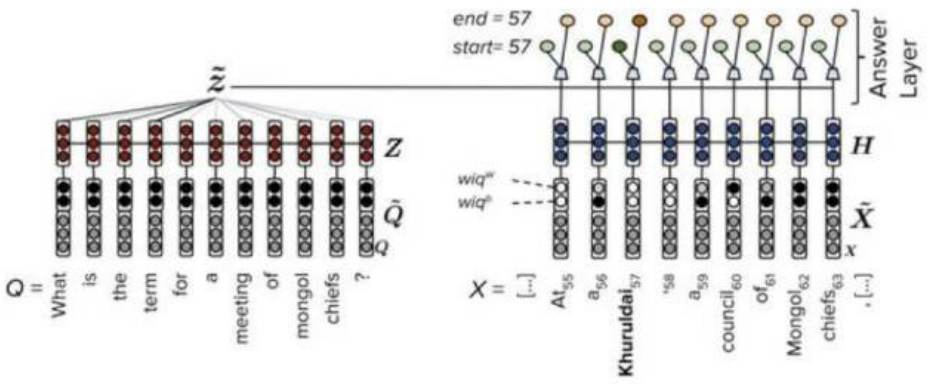
• FastQAExt [9]

FastQAExt 最大的特点在于其较为轻量级的网络结构，在输入的 embedding

层加入了两个简单的统计特征：

（1）文章中的词是否出现在问题中，是一个 binary 特征。

（2）基于“问题中的词如果在原文中很少出现，则对问题的回答影响更大”的原理，设计了一个 weighted 特征。



2 个统计特征的引入相当于给模型提前提供了先验知识，这将加快模型的收敛速度，整体上，FastQAExt 由以下三个部分组成：

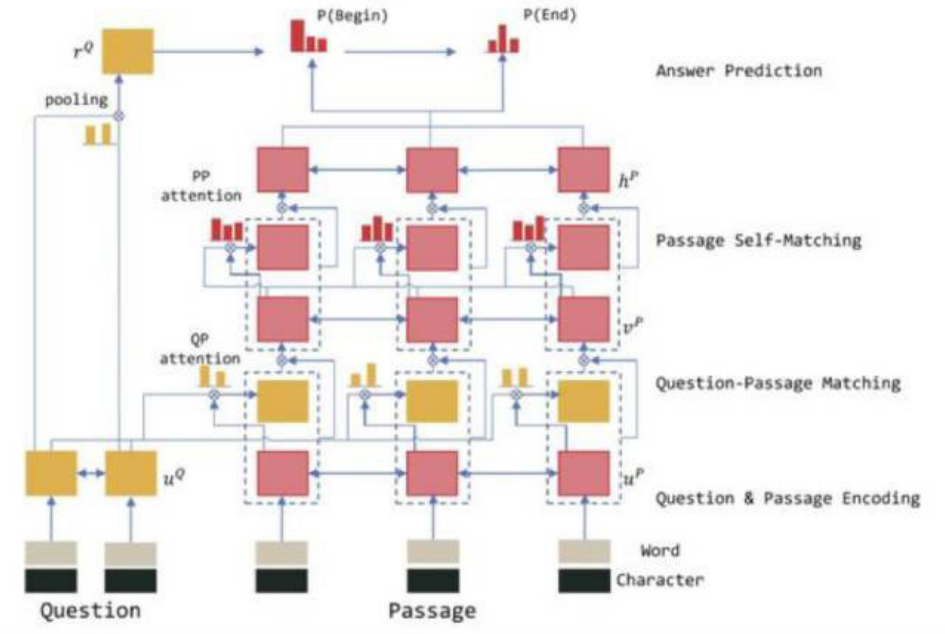
（1）Embedding 层：word 和 char 两种 embedding，且拼接上述的 2 种统计特征作为输入向量。

（2）Encoding 层：汇总原文和问题的总表示。

（3）Answer 层：计算问题对总表示，将 query-aware 原文表示和问题总表示共同输入两个前馈网络产生答案开始和结束位置概率。

● r-net [10]

r-net 是目前在 SQuAD LeaderBoard 上排名领先的模型，r-net 的特点是使用了双 interaction 层架构。



r-net 由以下三个部分组成：

（1）Encoding 层：使用 word 和 char 两种 Embedding 作为输入。

（2）Question-Passage Matching 层：负责捕捉原文和问题之间的交互信息。

（3）Passage Self-Matching 层：负责捕捉原文内部各词之间的交互信息。

（4）Answer 层：借鉴了 match-lstm 及 pointer network 的思路来预测答案的开始和结束位置，并在问题表示上用 attention-pooling 来生成 pointer network 的初始状态。

业务场景下的挑战与实践

通过上述介绍可以看到，围绕 SQuAD 数据集的机器阅读理解模型已经在学术界取得了相当大的突破，其解决的问题是在一定长度的 wiki 内容中进行知识问答，然而阿里小蜜的实际电商业务场景与之相比篇章更长、答案粒度更粗（业务场景下句子粒度居多）、业务含义更复杂且用户的提问更为随意，因此 SQuAD 数据集及其相关模型还不能直接运用于解决我们实际的电商场景问题。

要将机器阅读理解技术运用到实际业务场景中，还存在相当大到挑战，我们在以下几方面进行了探索和实践：

- 中文数据集的构建

为了使得模型能解决特定的业务问题，标注一个高质量的业务数据集是必不可少的，然而人工标注的成本较高，因此在业务数据集之外，需要将公开的数据集进行结合利用。而目前公开的中文数据集较为缺乏，可以通过批量翻译等方式快速构造中文数据集，翻译得到的结果由于保持了词汇及大致的上下文信息，也能取得一定的训练效果。

- 模型的业务优化

需要改进模型的结构设计，使得模型可以支持电商文档格式的输入。电商规则文档往往包含大量的文档结构，如大小标题和文档的层级结构等，将这些特定的篇章结构信息一起编码输入到网络中，将大幅提升训练的效果。

- 模型的简化

学术成果中的模型一般都较为复杂，而工业界场景中由于对性能的要求，无法将这些模型直接在线上使用，需要做一些针对性的简化，使得模型效果下降可控的情况下，尽可能提升线上预测性能。例如可以简化模型中的各种 bi-lstm 结构。

- 多种模型的融合

当前这些模型都是纯粹的 end-to-end 模型，其预测的可控性和可解释性较低，要适用于业务场景的话，需要考虑将深度学习模型与传统模型进行融合，达到智能程度和可控性的最佳平衡点。

总结

机器阅读理解是当下自然语言处理领域的一个热门任务。近年来，各类阅读理解的数据集以及方法层出不穷，尤其是围绕 SQuAD 数据集的模型正在快速的发展中，这些模型的研究在学术界非常活跃。总的来说，对于解决 wiki 类客观知识问答已经取得比较好的结果，但对于复杂问题来说仍处于比较初级的阶段。

学术界的思路和工业界实际场景相结合将能产生巨大的价值，阿里小蜜已经在这

方面开展探索和落地的尝试，在算法、模型和数据方面进行积累和沉淀，未来在更多的真实领域场景中，用户将能感受到机器阅读理解技术带来的更为便利的智能服务。

参考文献

[1] Weston et al. 2015. Towards AI-Complete Question Answering: A Set ofPrerequisite Toy Tasks
[2] Richardson et al. 2013. MCTest: A Challenge Dataset for theOpen-DomainMachine Comprehension of Text
[3] Hermann et al. 2015. Teaching Machines to Read and Comprehend
[4] Hill et al. 2015. The Goldilocks Principle: Reading Children's Books withExplicit Memory Representations
[5] http://hfl.iflytek.com/chinese-rc/
[6] Rajpurkar et al. 2016. SQuAD: 100,000+ Questions for Machine Comprehensionof Text
[7] Wang et al. 2016. Machine Comprehension Using Match-LSTM and Answer Pointer
[8] Seo et al. 2016. Bidirectional Attention Flow for Machine Comprehension
[9] Weissenborn et al. 2017. Making Neural QA as Simple as Possible but notSimpler
[10] Wang et al. 2017. Gated Self-Matching Networks for Reading Comprehensionand Question Answering

阿里妈妈首次公开自研 CTR 预估核心算法 MLR

阿里妈妈算法团队

一、技术背景

CTR（Click- Through-Rate）即点击通过率，是互联网广告常用的术语，指网络广告（图片广告 / 文字广告 / 关键词广告 / 排名广告 / 视频广告等）的点击到达率，即该广告的实际点击次数除以广告的展现量。点击率预估（Click-Through Rate Prediction）是互联网主流应用（广告、推荐、搜索等）的核心算法问题，包括 Google、Facebook 等业界巨头对这个问题一直进行着持续投入和研究。

CTR 预估是互联网计算广告中的关键技术环节，预估准确性直接影响公司广告收入。广告领域的 CTR 预估问题，面临的是超高维离散特征空间中模式发现的挑战——如何拟合现有数据的规律，同时又具备推广性。

二、CTR 预估算法现状及进展

2.1 传统 CTR 预估算法及不足

业界传统的 CTR 预估解法是广义线性模型 LR(logistic regression，逻辑斯特回归)+ 人工特征工程。LR 使用了 Logit 变换将函数值映射到 0~1 区间，映射后的函数值就是 CTR 的预估值。LR 这种线性模型很容易并行化，处理上亿条训练样本不是问题。但这种解法的不足是，因为线性模型的学习能力有限，需要引入大量的领域知识来人工设计特征以及特征之间的交叉组合来间接补充算法的非线性学习能力，非常消耗人力和机器资源，迁移性不够友好。

另外，目前业界也有一些效果不错的非线性模型不断被提出来，并被工程实践且取得不错效果，但这些模型都或多或少存在一些不足。比如 Kernel 方法，因为复杂度太高而不易实现；比如 Tree based 方法，这个是由 Facebook 团队在 2014 年首先提出，有效地解决了 LR 模型的特征组合问题，但缺点就是仍然是对历史行为的记

忆，缺乏推广性；还有 FM（factorization machine）模型，能自动学习高阶属性的权值，不用通过人工的方式选取特征来做交叉，但 FM 模型只能拟合特定的非线性模式，如最常用的 2 阶 FM 只能拟合特征之间的线性关系和二次关系。深度神经网络非线性拟合能力足够强，但面对广告这样的大规模工业级稀疏数据，适合数据规律的、具备推广性的网络结构业界依然在探索中，尤其是要做到端到端规模化上线，这里面的技术挑战依然很大。

那么挑战来了，如何设计算法从大规模数据中挖掘出具有推广性的非线性模式？

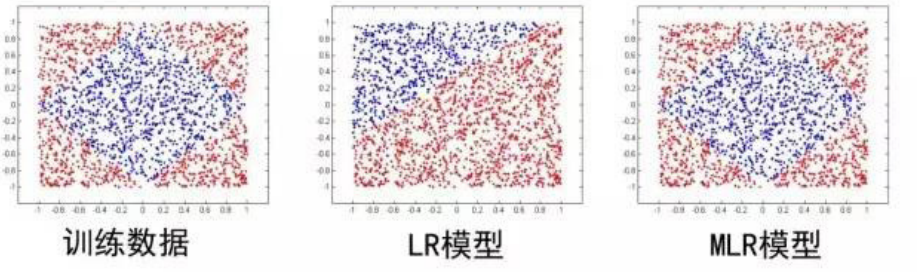
2.2 阿里妈妈自主研发 MLR 算法

2011-2012 年期间，阿里妈妈资深专家盖坤（花名靖世）突破了主流大规模线性模型的思路，创新性地提出了 MLR(mixed logistic regression, 混合逻辑斯特回归）算法，引领了广告领域 CTR 预估算法的全新升级。MLR 算法创新地提出并实现了直接在原始空间学习特征之间的非线性关系，基于数据自动发掘可推广的模式，相比于人工来说效率和精度均有了大幅提升。

MLR 可以看做是对 LR 的一个自然推广，它采用分而治之的思路，用分片线性的模式来拟合高维空间的非线性分类面，其形式化表达如下：

$$f(x) = \sum_{i=1}^{m} \pi_i(x) \cdot \eta_i(x) = \sum_{i=1}^{m} \frac{e^{\mu_i \cdot x}}{\sum_{j=1}^{m} e^{\mu_j \cdot x}} \cdot \frac{1}{1 + e^{-w_i \cdot x}}$$

这里面超参数分片数 m 可以较好地平衡模型的拟合与推广能力。当 m=1 时 MLR 就退化为普通的 LR，m 越大模型的拟合能力越强，但是模型参数规模随 m 线性增长，相应所需的训练样本也随之增长。因此实际应用中 m 需要根据实际情况进行选择。例如，在我们的场景中，m 一般选择为 12。下图中 MLR 模型用 4 个分片可以完美地拟合出数据中的菱形分类面。



MLR 算法适合于工业级的大规模稀疏数据场景问题，如广告 CTR 预估。背后的优势体现在两个方面：

1）端到端的非线性学习：从模型端自动挖掘数据中蕴藏的非线性模式，省去了大量的人工特征设计，这使得 MLR 算法可以端到端地完成训练，在不同场景中的迁移和应用非常轻松。

2）稀疏性：MLR 在建模时引入了 L1 和 L2,1 范数正则，可以使得最终训练出来的模型具有较高的稀疏度，模型的学习和在线预测性能更好。当然，这也对算法的优化求解带来了巨大的挑战，具体细节参见我们的论文（见文章尾部）。

2.3 MLR 算法高级特性

在具体的实践应用中，阿里妈妈精准定向团队进一步发展了 MLR 算法的多种高级特性，主要包括：

1）结构先验。基于领域知识先验，灵活地设定空间划分与线性拟合使用的不同特征结构。例如精准定向广告中验证有效的先验为：以 user 特征空间划分、以 ad 特征为线性拟合。直观来讲这是符合人们的认知的：不同人群具有聚类特性，同一类人群对广告有类似的偏好，例如高消费人群喜欢点击高客单价的广告。结构先验有助于帮助模型缩小解空间的探索范围，收敛更容易。

$$x = (x_1, x_2), \quad x_1 \in R^{d_1}, x_2 \in R^{d_2}, x \in R^{d_1+d_2}$$

$$f(x;\theta) = \sum_{j=1}^{m} \pi_j(x_1) \cdot \eta_j(x_2) \quad \longleftarrow \quad \text{结构先验/正则}$$

$$= \sum_{j=1}^{m} \frac{exp(\mu_j * x_1)}{\sum_{i=1}^{m} exp(\mu_i * x_1)} \cdot \frac{1}{1 + exp(-w_j * x_2)}$$

X1：聚类参数
决定空间的划分

X2：分类参数
决定空间内的预测

2）线性偏置。这个特性提供了一个较好的方法解决 CTR 预估问题中的 bias 特征，如位置、资源位等。实际应用中我们对位置 bias 信息的建模，获得了 4% 的 RPM 提升效果。

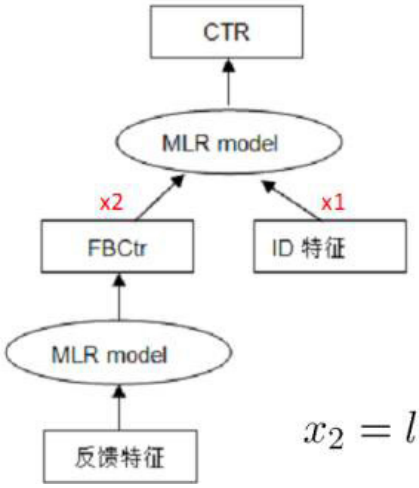
$$x = (x_1, x_2), \quad x_1 \in R^{d_1}, x_2 \in R^{d_2}, x \in R^{d_1+d_2}$$

$$f(x;\theta) \approx \left(\sum_{j=1}^{m} \pi_j(x_1) \cdot \eta_j(x_1) \right) \cdot \eta(x_2)$$

$$= \left(\sum_{j=1}^{m} \frac{exp(\mu_j x_1)}{\sum_{i=1}^{m} exp(\mu_i x_1)} \cdot \frac{1}{1 + exp(-w_j x_1)} \right) \cdot \frac{1}{1 + exp(-w x_2)}$$

- ✓ **Position bias**
  排名第1位和第5位的样本，点击率天然存在差异
- ✓ **Sample bias**
  pc和mobile上的样本，点击率天然存在差异

3）模型级联。MLR 支持与 LR 模型的级联式联合训练，这有点类似于 wide&deep learning。在我们的实践经验中，一些强 feature 配置成级联模式有助于提高模型的收敛性。例如典型的应用方法是：以统计反馈类特征构建第一层模型，它的输出（如下图中的 FBCtr）级联到第二级大规模稀疏 ID 特征体系中去，这样能够有助于获得更好的提升效果。



$$
\begin{aligned}
ctr &= f(x;\theta) \\
&\approx \left(\sum_{j=1}^{m} \pi_j(x_1)\cdot\eta_j(x_1)\right)\cdot\eta(x_2) \\
&= f_1 \cdot \frac{1}{1+exp(-wx_2)} \\
&= f_1 \cdot \frac{1}{1+exp(-w\cdot log(f_2))} \\
&\approx f_1\cdot(f_2)^w \\
&= f_1\cdot(FBCtr)^w
\end{aligned}
$$

$$x_2 = log(f_2) = log(\mathrm{FBCtr})$$

4）增量训练。实践证明，MLR 通过结构先验进行 pretrain，然后再增量进行全空间参数寻优训练，会获得进一步的效果提升。同时增量训练模式下模型达到收敛的步数更小，收敛更为稳定。在我们的实际应用中，增量训练带来的 RPM 增益达到了 3%。

$$x = (x_1, x_2), \quad x_1 \in R^{d_1}, x_2 \in R^{d_2}, x \in R^{d_1+d_2}$$

Step 1：结构化先验预训练

$$f(x;\theta_1) = \sum_{j=1}^{m} \pi_j(x_1)\cdot\eta_j(x_2) \longleftarrow \text{结构先验/正则}$$

$$= \sum_{j=1}^{m} \frac{exp(\mu_j * x_1)}{\sum_{i=1}^{m} exp(\mu_i * x_1)} \cdot \frac{1}{1+exp(-w_j * x_2)}$$

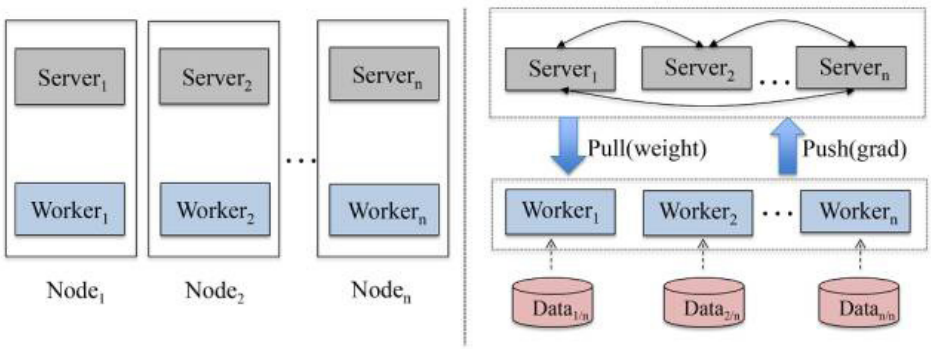
Step 2：以 θ_1 为初值在全空间增量化训练

$$f(x;\theta_2) = \sum_{j=1}^{m} \pi_j(x)\cdot\eta_j(x)$$

θ_2 初始化为 θ_1

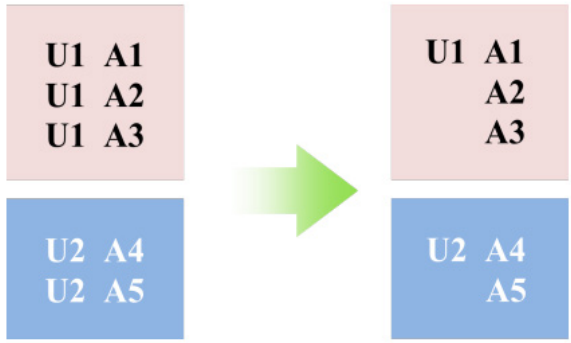
2.4 大规模分布式实现

MLR 算法面向的是工业级的数据，例如亿级特征，百亿级参数，千亿级样本。因此我们设计了一套分布式架构，以支持模型的高效并行训练。下图是架构示意图，它跟传统的 parameter server 架构略有区别，主要不同点在于我们在每一个分布式节点上同时部署了 worker 和 server 两种角色，而不是将 server 单独剥离出去部署。这背后的考虑是充分利用每个节点的 CPU 和内存，从而保证最大化机器的资源利用率。



A) Distributed Topology　　B) model-parallelism and data-parallelism

此外，针对个性化广告场景中数据的结构化特性，我们提出并实现了 common feature 的 trick，可以大幅度压缩样本存储、加速模型训练。例如下图示意，在展示广告中，一般来说一个用户在一天之内会看到多条广告展现，而一天之内这个用户的大量的静态特征（如年龄、性别、昨天以前的历史行为）是相同的，通过 common feature 压缩，我们对这些样本只需要存储一次用户的静态特征，其余样本通过索引与其关联；在训练过程中这部分特征也只需要计算一次。在实践中应用 common feature trick 使得我们用近 1/3 的资源消耗获得了 12 倍的加速。



三、MLR 在阿里妈妈业务应用现状

从 2013 年起，MLR 算法在阿里妈妈及阿里集团多个 BU 的主要场景（包括阿里妈妈精准定向广告、淘宝客、神马商业广告、淘宝主搜等等）被大规模地应用和尝试，尤其是在阿里妈妈的精准定向广告场景，算法模型创新带来了业务上的重大突破，主要场景下的 CTR 和 RPM 均获得 20% 以上的提升。典型应用如下：

3.1 基于 MLR 的定向广告 CTR 预估算法

基于 MLR 算法的非线性学习能力，阿里妈妈的定向广告 CTR 预估采用了大规模原始 ID 特征 +MLR 算法的架构。具体地，我们刻画一次广告展现为特征向量，它由三部分独立构成：用户部分特征（包括 userid、profile 信息、用户在淘宝平台上的历史行为特征（浏览 / 购买过的宝贝 / 店铺 / 类目上的 id 和频次等）、广告部分特征（包括 adid、campainid、广告对应的卖家店铺 id、类目 id 等）、场景部分特征（包括时间、位置、资源位等）。这些特征之间无传统的交叉组合，维度在 2 亿左右。然后我们将数据直接喂给 MLR 算法，并且应用了结构化先验、pretrain+ 增量训练、线性偏置等高级技巧，让模型从数据中自动去总结和拟合规律。实践证明，相比于传统的 LR+ 特征工程思路，这种解法更为高效和优雅，模型精度更高，在实际生产中的可迭代更强。

3.2 基于 MLR 的定向广告 Learning to Match 算法

Match 算法是定向广告中的一个重要环节，它的核心使命是基于用户的人口属

性、历史行为等信息来猜测用户可能感兴趣的广告集合。传统的 Match 算法更多采用的是规则匹配、协同过滤等方法，方法的扩展性不强。在阿里妈妈定向广告系统中，我们研发了基于 MLR 的 learning to match 算法框架。简单来说，用模型的方法基于用户的行为历史来学习用户个性化的兴趣，从而召回高相关性的候选广告集。同样地，基于 MLR 算法的非线性能力，我们可以很容易地将不同的特征源、标签体系融合到框架中，不需要过多地关注和设计特征的交叉组合，使得框架的灵活性大大增强。

四、总结和挑战

总的来说，阿里妈妈算法技术团队自主创新的 MLR 模型和算法，在阿里妈妈业务中大范围推广和应用带来了非常好的效果，另外在大数据智能方面，因为省去特征工程，具备了从数据接入到应用的全自动功能。

虽然目前取得了非常不错的成绩，但是未来的挑战也不小：比如初值问题、非凸问题的局部极值、虽然 MLR 比 LR 好，但不知道和全局最优相比还有多远；第二，在初值的 Pre-train 方面需要改进和优化模型函数等等；第三，目前规模化能力方面也需要能够吞吐更多特征和数据，比如采用更快的收敛算法等等；最后，整体的 MLR 算法的抽象能力也需进一步得到强化。

关于 MLR 算法的详细技术细节，包括建模思路、优化算法、大规模并行高效实现等等，我们在 arxiv 上的最新文章 (https://arxiv.org/abs/1704.05194) 中进行了披露（注：论文中为了严谨性使用 LS-PLM: Large Scale Piecewise Linear Model 来命名 MLR 算法）。

阿里翻译一年 2500 亿次调用，节省 25 亿美元

寻则　品取　忱梦

神经网络机器翻译 (Neural Machine Translation, NMT) 模型自 2013 年在学术界首次被提出后，就不断快速发展，目前在某些语种和场景下，译文质量甚至可以达到人工翻译的水平。

阿里翻译团队自 2016 年 10 月起正式开始自主研发 NMT 模型，2016 年 11 月首次将 NMT 系统的输出结果应用在中英消息通讯场景下的外部评测中并取得了不错的成绩，翻译质量有了大幅度提升。

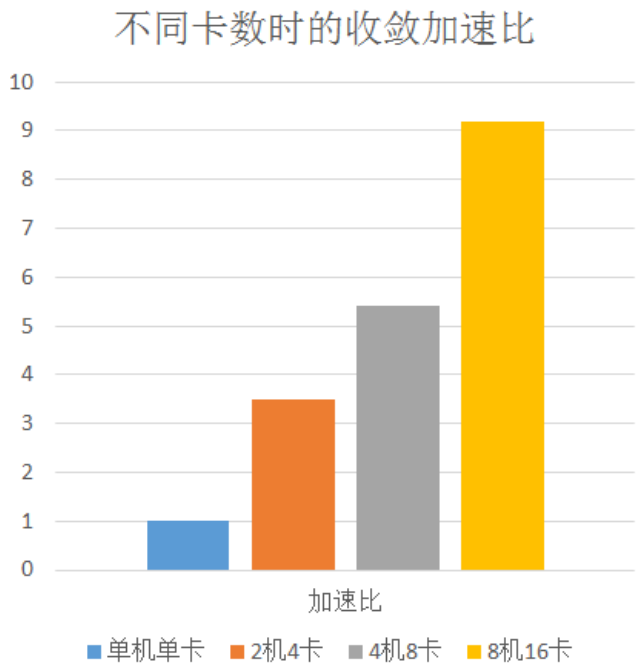
但是，由于 NMT 模型的结构复杂，且深度神经网络模型本身的训练过程一般又会涉及很大量的计算，因此 NMT 系统往往需要较长的训练周期，例如，使用 3000 万的训练数据在单块 GPU 卡上一般需要训练 20 天以上，才能得到一个初步可用的模型。

基于上述问题，2017 年 2 月初开始，阿里翻译团队和阿里云 Large Scale Learning 的穆琢团队合作，共同开发支持分布式训练的 NMT 系统，并于 2017 年 3

月底完成了第一个版本的分布式 NMT 系统。

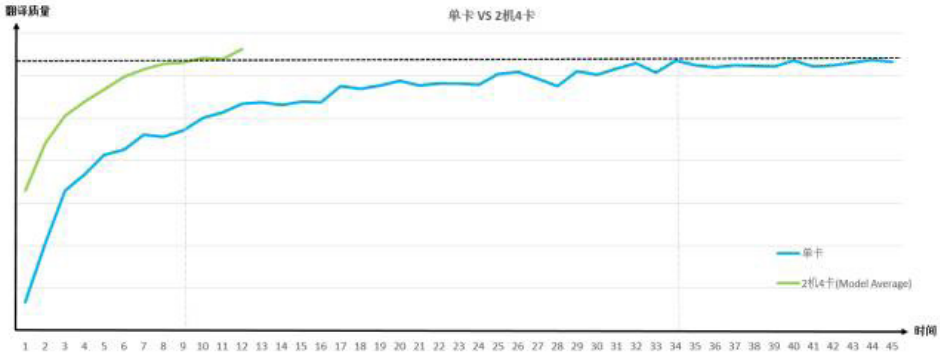
项目成果

在 2017 年 4 月份的英俄电商翻译质量优化项目中，分布式 NMT 系统大大提高了训练速度，使模型训练时间从 20 天缩短到了 4 天，为项目整体迭代和推进节省了很多时间成本。

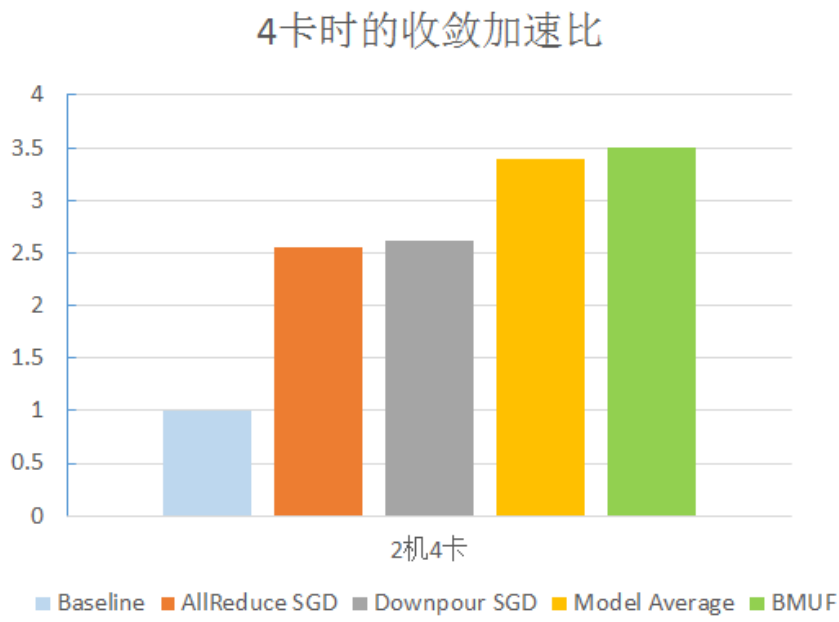


图一：使用不同卡数时，在中英 100 万训练语料上获得的收敛加速比

在 4 张 GPU 卡上，可以达到 3 倍以上的收敛加速比，在 8 张 GPU 卡上，可以达到 5 倍以上的收敛加速比，在 16 张 GPU 卡上，可以达到 9 倍以上的收敛加速比，通过不断累加 GPU 卡数，收敛加速比预期还会继续提升。



图二：中英 2000 万训练集上，利用 2 机 4 卡进行分布式训练的收敛过程



图三：多种分布式策略的加速效果对比

除了基于 MA 的分布式实现，项目组还尝试了其他多种分布式实现方法，也都获得了不同程度的加速效果，包括 Downpour SGD、AllReduce SGD 以及使用了 BMUF(Blockwise Model-Update Filtering, 一种针对 Model Average 方法的改进方案) 策略的 Model Average 方法。

实现方案

关于多机多卡分布式方案的讨论

• data parallel 数据并行

基于 data parallel 的同步 SGD 算法，即使用多个 worker 均摊整个 batch 的计算量，且每个 GPU（worker）上存储完整的模型副本。 Tensorflow 本身提供的 PS 框架可以直接用于实现此算法。标准的同步 SGD 算法每次迭代都分为三个步骤，首先，从参数服务器中把模型参数 pull 到本地，接着用新的模型参数计算本地 mini-batch 的梯度，最后后将计算出的梯度 push 到参数服务器。参数服务器需要收集所有 workers 的梯度，再统一处理更新模型参数。

因此，在 ps 框架下，同步 SGD 算法的总通信量 transfer data = 2 * num_of_gpus * size_of_whole_model。通过对模型的大小和一个 mini-batch 计算时间的评估，发现在单机两卡的情况下，由于 PCIe 的带宽较高（~10GB/s），可以获得接近 2 倍的加速比，但两机两卡的情况下，计算通信比则为 1:1，而随着机器数的增多，不但没有带来加速，反而比单卡更慢。

可见，通信优化是同步 SGD 算法的主要问题，下面的具体方案介绍中，我们也展示了相关的实验结果。

• hybrid parallel 混合并行

模型并行可以通过模型参数的存储策略，利用参数的稀疏性，减少参数的实际通信量，对训练过程进行加速。一般情况下，适用于稀疏模型的训练，但对于特殊的网络结构，模型并行可以和数据并行相结合，提高全局的计算通信比，我们这里称之为 hybrid parallel。以 AlexNet 举例来说，可以把模型结构分为全连接层和卷积层两部分，从而对两个部分分别分析计算通信比。

全连接层计算通信比极低，毫无扩展性，而卷积层的计算通信比较高，适合使用数据并行，因此 hybrid parallel 的方案是：多个 workers 使用数据并行的方式计算卷积层，而只需一个 worker（或极少量 workers）计算全连接层，以获得最高的计算通信比，且卷积层和全连接层中间只有一个规模很小的 feature map 需要通信。

通过对 NMT 的模型中各个层进行 profiling，发现网络中各部分的参数大小相差

不大，计算量也相当，不但计算通信比不高，各层结构之间传递的 feature map 也较大，并不适合使用 hybrid parallel 的方式。

• 分布式方案的进一步探索

基于以上的分析，我们对分布式方案进行了进一步的探索。为了最大程度的获得多机多卡的扩展性，我们选择了 Model average、BMUF、downpour 等分布式方法。Model average（MA）方法较于简单的数据并行，有两个显著的优势：一、通信量可以通过同步频次来调节，两次同步的间隔可以以 step 甚至 epoch 为单位来控制，一般情况下，通信开销几乎可以忽略不计，从而获得线性的计算加速比；二、batch size 的大小可以和单机 baseline 保持一致，基于数据并行的同步 SGD，通常为了保持计算通信比而同比增大 batch size，导致精度损失。

但是，MA 方法、BMUF 方法以及基于异步更新的 downpour 方法，对调参的要求都更高，不同的参数设置，会对模型的收敛结果有较大的影响，计算加速比的提高并不意味着收敛加速比的提高。下文是对各个实现方案的消息介绍和结果展示。

基于 Model Average 方法的分布式训练方案

• 方案概述

Model Average 方法，即每个 worker 负责训练本地的模型，多个 GPU 之间通过固定（或动态）的频率求取各个模型的均值，并以此作为继续训练的基础。

基于 Tensorflow 的 Model Average 设计主要分为 graph 构建和分步 run 两部分。

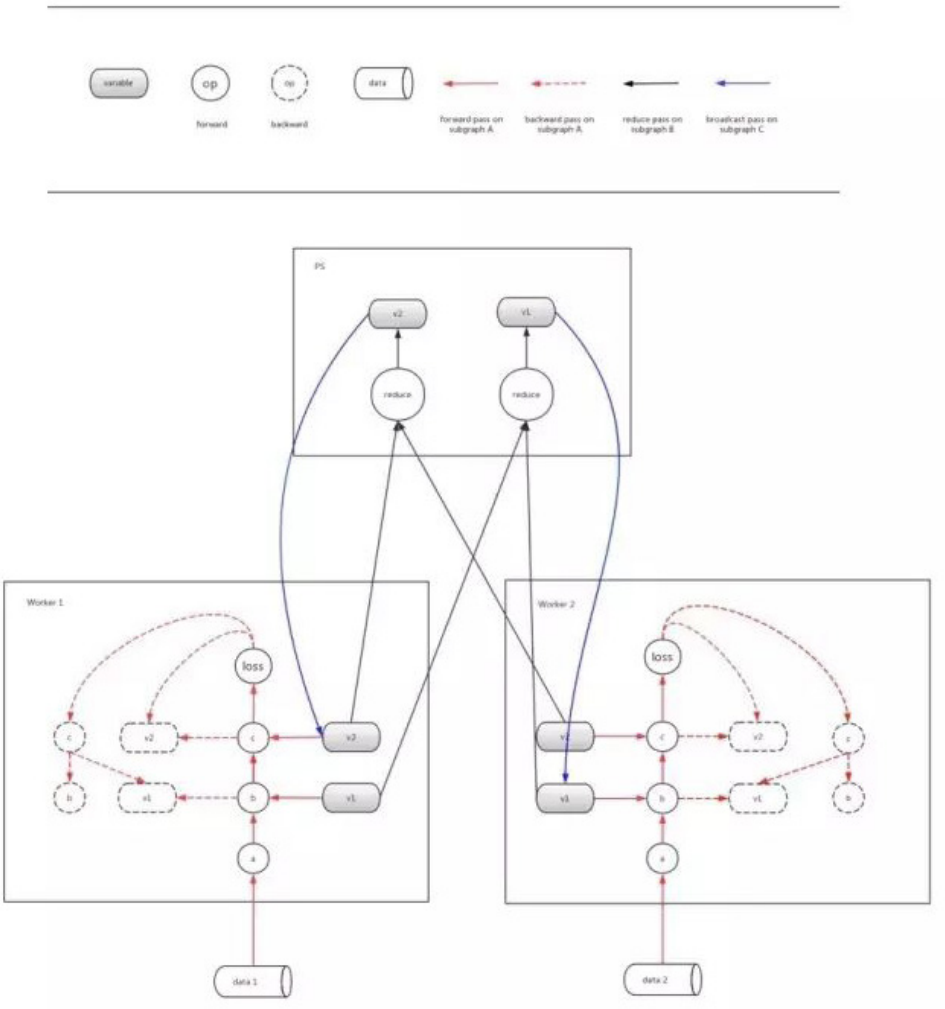
1. 首先，由于每个 worker 需要在各自的进程内进行求解，所以需要在每个 worker 上分别进行构建 forward-backward 子图，这一部分的子图在上图中用红色的箭头线表示。应当注意，每个 worker 上应该具有独立的 op 和 variable。

2. 其次，需要在 ps 上维护一份模型拷贝，并构建 Reduce Average 的子图，这一部分的子图在上图中用黑色的箭头线表示。

3. 然后，需要构建 broadcast 的子图，这一部分的子图在上图中用蓝色的线表示。（这里由于美观，在图中没有画出全部的 broadcast 依赖关系，ps 的 variable 1 和 variable 2 都应该和 worker 1 和 worker 2 中的 variable 1 和 variable 2 建立依赖关系）

4. 在构建上述依赖关系后，我们需要通过 client 启动 Model Average 的机制。用户只需要通过控制 session run 的顺序来指定每个子图的求解即可完成。在这里，红色的子图需要每轮都 run，而到了一定轮数间隔后开始依次 run 黑色的子图和蓝色的子图即可完成 Model Average 的机制。

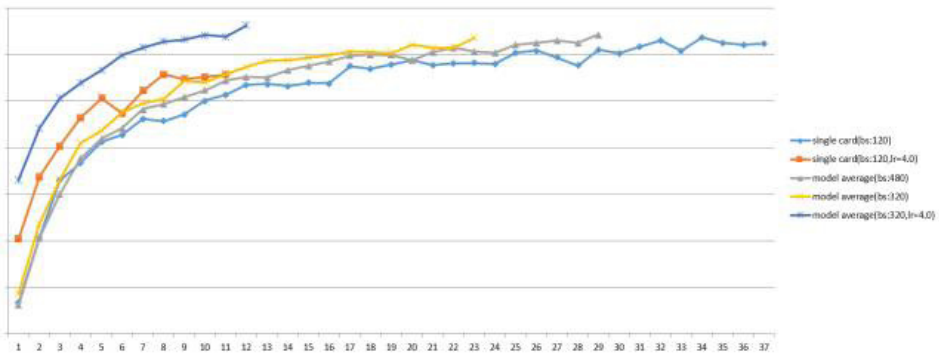
在实现上，可以用主 worker 完成所有子图的构建，其他 worker 处于 join 状态。在运行时，各个 worker 会根据图中的依赖关系驱动起来。



- 超参的调整

当两机四卡的参数保持和单机单卡 baseline 参数一致的情况下，从多组实验中可以观察到，不同的同步频次对模型的质量有影响，加速比最快只达到 1.5。 通过推导，我们发现，MA 方法中每次计算多个 worker 的模型均值，都相当于把每个训练样本对模型梯度的贡献缩减为原来的 1/num_of_gpus, 大大抵消了多卡带来的加速。因此，MA 中的本地训练 learning rate（lr）应该调整为原 baseline lr 的 num_of_gpus 倍。

- 实验结果



其中 bs 表示 batch size，lr 表示 learning rate，横坐标是训练时间，纵坐标代表翻译质量。

使用 BMUF 方法对 Model Average 方案进行改进

- 方案概述

上面提到的 Model Average 方法在两卡、四卡、八卡上均获得了较高的加速比，在不断并行扩展的过程中，需要根据机器的个数对 lr 进行调整，然而 lr 的大小一定有一个适当的范围，并不能无限的增大，随着机器数的增多，这种 naive MA 方法也逐渐暴露出它的局限性。

BMUF（Blockwise Model-Update Filtering）是 Model average（naive MA）的一种优化算法，最主要的区别是引入了梯度的历史量。MA 算法等价于在每次同步时，把所有 workers 上的模型增量加和平均，作为“block gradients”，用于更新参

数服务器上的全局参数，而 BMUF 的“block gradients”则在此基础之上，增加了另外一项历史梯度，且历史梯度的权重为 block momentum rate。

具体推导如下：

γ is the learning rate.
ϵ is the momentum rate.

ζ is the block learning rate.
η is the block momentum rate.

MA：一个 mini-batch 的训练数据对最后模型的贡献公式为：

$$\delta^{(i)}{}_m = \frac{1}{N}\gamma_m g^{(i)}{}_m(1+\epsilon_m+\epsilon_m{}^2+\ldots+\epsilon_m{}^{\tau-l}) \approx \frac{1}{N}\frac{\gamma_m(1-\epsilon_m^{\tau-l+1})}{1-\epsilon_m}g^{(i)}{}_m$$

BMUF：一个 mini-batch 的训练数据对最后模型的贡献公式为：

$$\delta^{(i)}{}_b = \frac{1}{N}\frac{\gamma_b(1-\epsilon_b{}^{\tau-l+1})}{1-\epsilon_b}g^{(i)}{}_b\zeta(1+\eta+\eta^2+\ldots) \approx \frac{\zeta}{N(1-\eta)}\frac{\gamma_b(1-\epsilon_b{}^{\tau-l+1})}{1-\epsilon_b}g^{(i)}{}_b$$

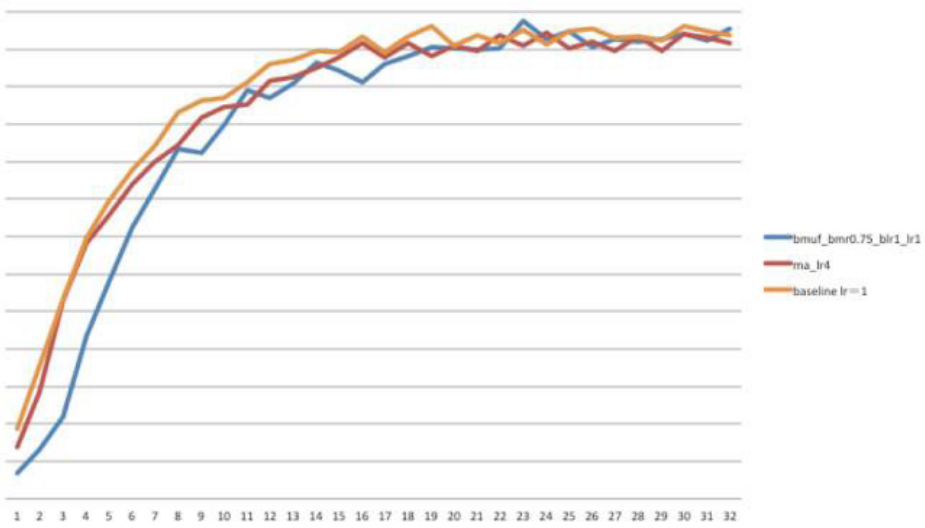
从公式中可以看出，MA 中单个 mini-batch 的贡献在 average 阶段被缩小为原来的 N 分之 1，如果要保持和 baseline 一样的贡献度，则需要增大 lr。而 BMUF 相比 baseline 的贡献比为：

$$\frac{\zeta}{N(1-\eta)}$$

- 超参的调整

上文的公式可以指导 BMUF 训练过程中参数的调整。BMUF 最主要的参数有两个，一是 block learning rate（blr），此超参一般设为 1.0，和 MA 算法一致，二是 block momentum rate（bmr），可以通过调整 bmr 为 1-1/N，从而保持 lr 不变，总的贡献量和 baseline 一致。此外，引入了历史梯度后，还可以结合 Nesterov 方法，在本地训练时先对本地的模型更新已知的历史梯度，再进行训练，从而进一步加速了收敛。

- 实验结果



上图横坐标代表已训练的总数据量，即每个计算节点训练数据量的总和（最理想的情况，收敛曲线应该和 baseline 重合甚至更高），纵坐标代表模型的翻译质量。可以看到 MA 和 BMUF 在两机四卡上可以获得相当的加速比，BMUF 的优势在于，可以通过调节 bmr 从而把 learning rate（lr）可以保持在与单机单卡的 baseline 一个量级，因为 lr 有一个较为适当的范围，不能随着 GPU 的增多无限制增大。BMUF 的效果在更大的 lr 以及十六卡的实验中已得到体现。

基于 Downpour SGD 方法的分布式训练方案

- 方案概述

和 MA 的设计类似，Downpour SGD 的设计也比较自然，同样分为 graph 构建和分布 run 两部分。在这里需要注意的是，Downpour SGD 方法是一种 ASGD 的异步分布式训练方法，其在 Apply gradients 时是不需要多个 worker 之间同步的。

下面具体介绍 Downpour 的整体流程。

1. 将 ps 上的 model weights 拉到每个 worker 上。

2. 每个 worker 自己求解本地的 model，在求解过程中，不但要对本地 model 进行参数更新，还要将梯度累加到另一组 variable 中保存。

3. 当本地 worker 求解到一定轮数后，将本地存储的累积梯度异步 apply 到 ps 的 model 参数上。

这种方式在实现上，需要在每个 worker 上再保存一份 variable 用来存储梯度的累积量，并且在构图时需要对 optimizer 进行更改，主要体现在需要将 AdaDelta 计算的梯度在 apply 到自己的模型上之前先累加到本地的 variable 中，然后再 apply 到自身参数中。

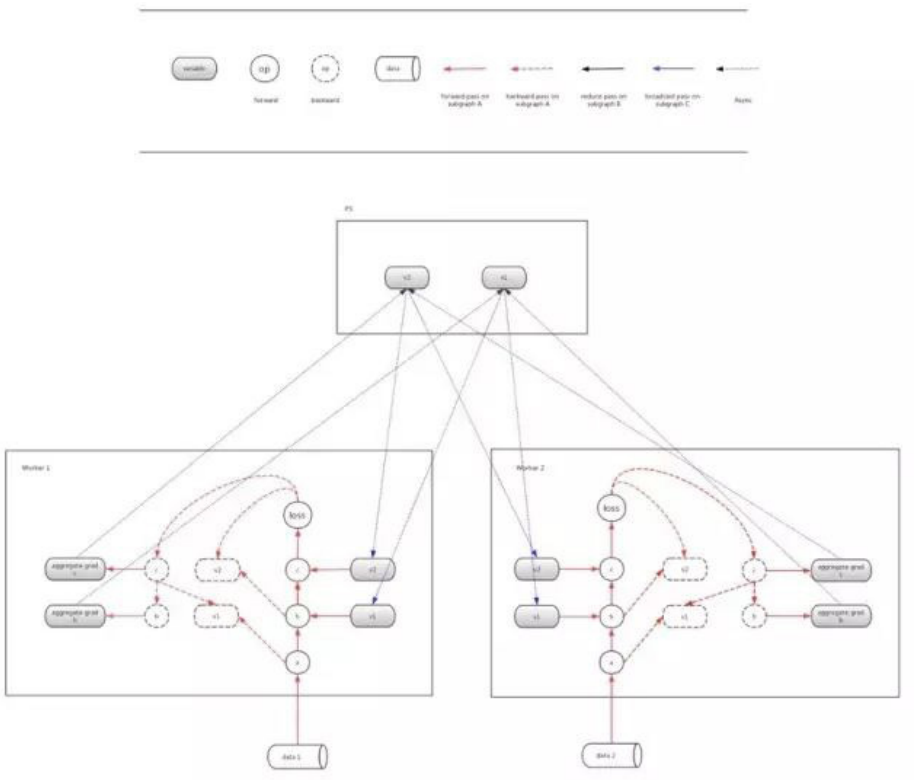
这里还存在一个问题关于累加梯度的问题，因为在 AdaDelta 的实现中，实际上 apply 到 model 上的量包含梯度和 delta 量两个部分，因此在累加上有可能需要将这些 delta 量也考虑进去。在实现时，无需重写新的 optimizer，只需要将调用 apply gradients 前后的 model weights 相减即可得出正确的结果。

Downpour SGD 会带来一些其他的超参数，这其中就包括 push gradients 和 pull weights 的间隔轮数，我们称之为 step。另外，若 push 和 pull 的 step 过大，也会导致这期间的累积梯度过大，如果此时直接将累积梯度 apply 到 weights 上很可能会出现 NAN 的情况，因此需要对累积梯度做 clipping 操作，所以对累加梯度进行 clipping 的 norm 值也是一个额外的超参数。

- 超参的调整

由于 ASGD 带来的异步性，导致调参的难度相对于同步 SGD 更为困难，而且 Downpour SGD 的超参数确实比较多。为了加快训练的速度，减少每个 epoch 的迭代时间，我们采用了比较激进的 weak scaling 的方式，即每个 worker 都使用较大的 batch size。

但是应当注意，虽然增加 batch size 会使过数据的速度加快，但也会让每个数据的学习效果降低，这是因为和小 batch size 相比，我们更新模型的频率大大降低了，因此我们需要同时加大 learning rate。另外，在异步条件下，每个 worker 计算的累积梯度都有可能成为 stale 的 gradients，这些 gradients 的表示的方向并不一定最优，相反，有时候会很差。



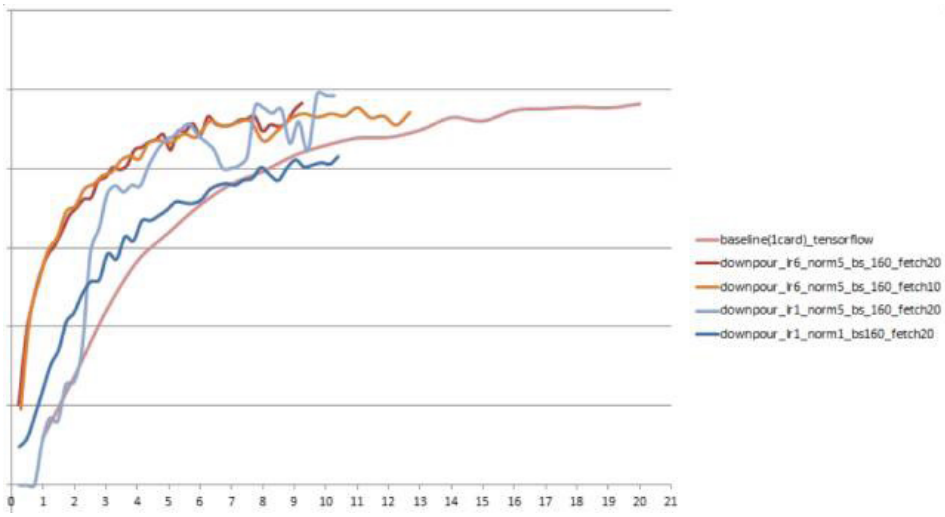
倘若我们不对这些累积的梯度进行 clip，那么很可能出现 NAN 的情况，因此我们需要调整累积梯度的 clipping norm 值。但是 clipping norm 值越小，梯度值就被削弱的越狠，模型的学习效果就越不好，它与 learning rate 的调参是一对 trade off。最后，用于 push 和 pull 的 step 间隔值不宜过大，因为过大的 step 会导致过大的累积梯度，如果此时使用较小的 clipping norm 对累积梯度进行削减，那么这将意味着如此长轮数计算出来的累积梯度效果将被削减。在实验中我们发现这个 step 间隔轮数设置在 20 左右比较理想。

综上所述，调参的基本目标是，固定 push 和 pull 的 step 间隔轮数后，选取较大的 batch size 并适当增加 learning rate，逐步调整增加 clipping norm 的大小，使学习效果达到最大。

- 实验结果

其中 bs 表示 batch size，lr 表示 learning rate，norm 将累积梯度做 clipping

的最大值，fetch 为 downpour 中 push 和 pull 的间隔值，纵坐标代表翻译质量值，横坐标是训练时间。



基于 Ring-allReduce SGD 方法的分布式训练方案

• 方案概述

上文提到，同步 SGD 算法每次迭代都需要 pull 参数和 push 梯度这两次通信，这样的一个 push + pull 的过程对于整个系统来说，等价于一次 allreduce 操作，通信量相当于 2 * nGPUs * model_size，随着 GPU 个数的增多，allreduce 成为最主要的性能瓶颈。

• Ring-allReduce op

Allreduce 聚合通信已有较成熟的优化方法，最常用的方法之一有 Ring allreduce。其基本思想是将 allreduce 分拆成 reduce_scatter 和 allgather 两步。

1. 首先，根据节点数对待通信的 message 进行分片，分片的数量与通信的节点数相同（不需要额外的参数服务器）。

2. 开始进行 reduce_scatter 操作。对 worker(i) 来说，第 j 次通信是把自己的第 j 片 message 发送给 worker(i+1)，并把收到的消息累加到 message 的对应片段。如此经过 n-1 轮的环状流水线通信后，每个 worker 上都有一个分片是 reduce

了所有 workers 的结果。

3. 然后开始 allgather 操作。第 j 次通信是把自己的第 j+1 片 message 发送给 worker(i+1)，如此经过 n−1 轮通信，所有 worker 的整个 message 就都经过了 reduce 操作，即完成了 all_reduce 操作。

ring-allreduce 方法可以使得集群内的通信总量保持在一个常数，略小于 2 * mdoel_size，不随并行规模的增大而增多，有很好的扩展性。

• gRPC vs MPI

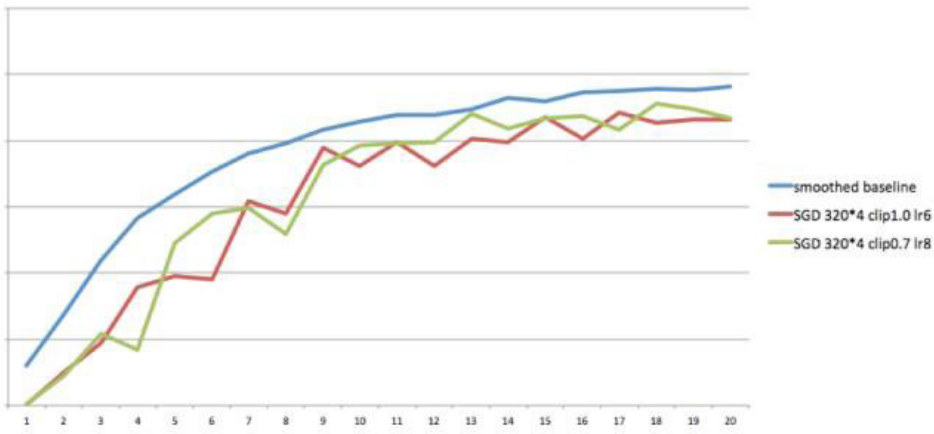
Ring- allreduce 方法能够带来性能提升的前提是：latency 的开销相对于 bandwidth 来说可以忽略不计，因为 ring allreduce 虽然降低了通信总量，却增加了通信次数。当前 tensorflow 的通信协议只支持 gRPC，通信的 latency 较高不能被简单忽略，而使用基于支持 RDMA 的 cuda-aware MPI 来实现 allreduce op，效果则更加显著。

此外，allreduce 操作同样存在于 Model average 算法中，同样可以带来一定的性能收益。

• 超参的调整

因为对于同步 SGD 分布式算法，如果保持 Total Batch size 不变，每块 GPU 卡上的 mini batch size 会随着 GPU 卡数的增多而减小，其计算时间并不是线性减少，因为当 batch size 足够小后，计算时间会逐渐趋于平稳，虽然通信已经通过优化而大幅减少，计算的拓展性依然有限。因此，在使用多机多卡的同步 SGD 时，batch size 与 GPU 个数同比例增大，同理，为了保持单个训练样本的贡献，lr 也同比增大。

- 实验结果



上图中，多卡同步 SGD 的总 batch size 设置为 1280，是 baseline(batch size=160) 的 8 倍，横坐标代表已训练的总数据量，纵坐标代表模型的翻译质量。在 InfiniBand（10GB/s）网络上，获得 4 倍的计算加速比，在万兆（1GB/s）以太网上，获得 2.56 倍计算加速比。但 batch size 的增大会影响收敛精度，上图可以看到，收敛的最高点比 baseline 略低，有明显的精度损失。

未来工作

上一阶段工作主要集中在模型训练阶段的加速策略上，接下来的工作主要分为两方面：一方面是继续挖掘分布式训练的加速潜力，通过系统与算法相结合的优化策略，最大化利用硬件资源，提升收敛加速比，并将分布式优化策略和算法模型本身解耦，实现复杂 DL 模型分布式加速功能的组件化和通用化。

另一方面，需要在现有的服务化方案的基础上，进一步通过模型精度压缩、网络结构简化等方式，在保证模型效果的同时，提高解码速度，降低线上延时，进而增强线上服务能力，节约服务化所需的硬件成本。

战胜柯洁后，AI 在悄悄潜入人类下一个智慧堡垒

德衡

近日，在 QCon 北京 2017 上，来自阿里巴巴认知计算实验室的技术专家龙海涛（花名德衡）发表了题为《星际争霸和 AI》的专题演讲。《星际争霸》一直是游戏玩家心目中即时战略类的经典之作，历时十多年而不衰。

而如今它更成为深度强化学习、人工智能算法研究的一个主要平台和工具。因为其蕴含了多智能体协作、多任务学习、宏观策略规划等复杂问题，一旦取得部分突破和进展，对商业和社会发展都会带来极大影响。如国外的 DeepMind、Facebook 等公司相继投入大量人力基于它进行通用人工智能的研究。

在本次演讲中，德衡重点介绍了阿里巴巴如何在《星际争霸》游戏环境中研究人工智能算法，分享阿里在这方面研究尝试中得到的初步成果，并重点阐述多智能体协作在微观战斗场景中的应用，以及未来在这个平台上的进一步的研究方向等。

以下为演讲全文：

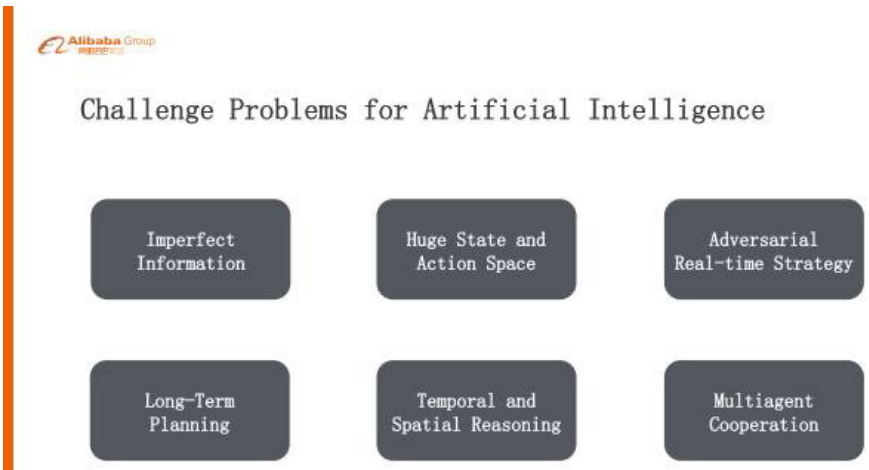
大家下午好！我是来自阿里巴巴认知计算实验室的龙海涛，今天主要跟大家聊一下“《星际争霸》与人工智能”的话题。首先我会介绍一下为什么会选择《星际争霸》

这个游戏来做人工智能前沿性的研究。

1. 为什么选择《星际争霸》作为人工智能算法研究的环境

首先可能大家有疑问，为什么选择《星际争霸》这个游戏来做我们 AI 研究平台。阿里认知计算实验室目前是挂靠在搜索事业部下面，我们团队的成员基本都是做搜索、广告、推荐、算法这样的背景，之前我们主要做的是 CTR 预估的优化，还有 CVR 转化率的一些优化，从去年“双 11”之后，我们想在认知智能方面做一些前沿性的探索，我们一致认为游戏是一个研究 AI 算法的绝佳平台，首先它是非常干净的平台，可以源源不断产生数据，而且迭代非常快，就是说它的智能是可以观测到的。

另外，它离真实的场景和应用比较近，并且《星际争霸》十多年来就是一个非常好的受大家欢迎的游戏，积累了非常非常多的数据，这样我们可以从之前的经验去学习，这也是我们考虑的一个方面。最重要的，它对 AI 来讲存在着非常大的挑战，非常复杂，主要有以下六点：



第一点，它是一个不完全信息下的环境。比起像围棋或者象棋这种大家都可能看得见的、完全信息下的博弈，《星际争霸》是有战争迷雾的，所以必须去探路、侦查、了解对手的信息，从而在不确定的情况下去做智能的决策，这个是相对其他游戏来讲非常不同或者挑战更大的一个方面。

第二点，它有非常巨大的搜索空间，围棋的搜索空间大概在 10^170，《星际争霸》在 128 × 128 的地图上并且人口上限是 400 个 unit 的情况下，它的搜索空间大概在 10^1685，比围棋高 10 个数量级，这还是在没有算上其他状态（比如说血量等等）的情况下。所以现有的任意一个单一的算法是根本不可能解决《星际争霸》里面所有的问题的。

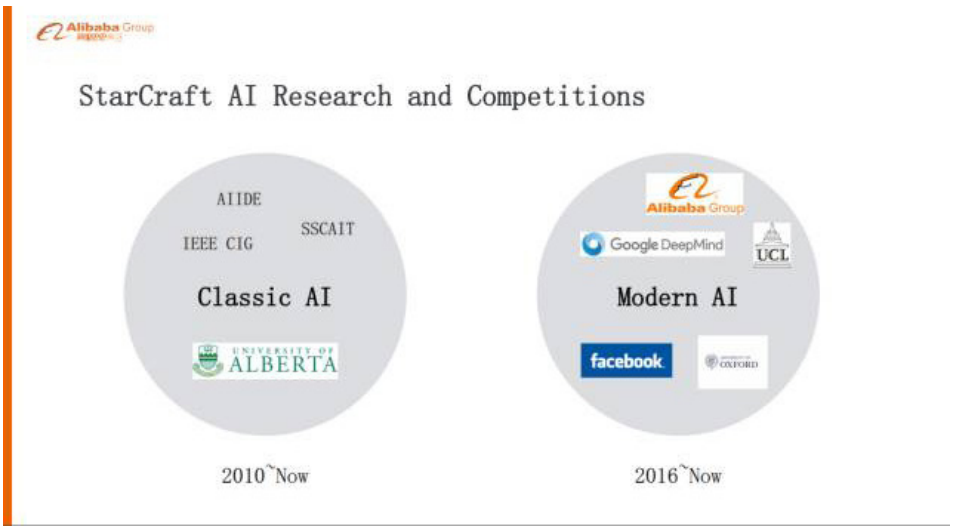
第三点，它是一个即时对抗类的游戏。下围棋可以有一分钟或者两分钟的思考时间，但是在《星际争霸》里，如果说正常游戏大概是 1 秒钟 24 帧，那么你必须在 42 毫秒之内做出迅速的反应，而且这个反应不是一个 action，而是一系列的 action，每个 unit 都会采取行动，这对我们算法的性能、效率、工程上的考虑都是非常大的挑战。

第四点，它需要智能体有一个长期的规划，而不是一个下意识的动作，是需要有记忆，需要考虑这场战争应该采取什么样的策略，中盘应该怎么考虑，发展到后期又应该采取什么样的策略，而且这个策略的计划是根据侦查到的所有的信息动态去调

整，这对人工智能的挑战是非常非常大的。

第五点，在《星际争霸》里面要玩好的话，必须基于时序上、空间上去做推理，比如说地理位置的优势，坦克如果架在哪里可能会比较好，如果开分机在哪个位置去开会比较有利，甚至于军营造在什么地方，这些对于 AI 来说都需要进行一个空间上的推理。

第六点，《星际争霸》最高有 400 个 unit，所以其实是需要多个智能体协作的，需要多个兵种去配合，这也是对 AI 来讲一个很大的挑战。



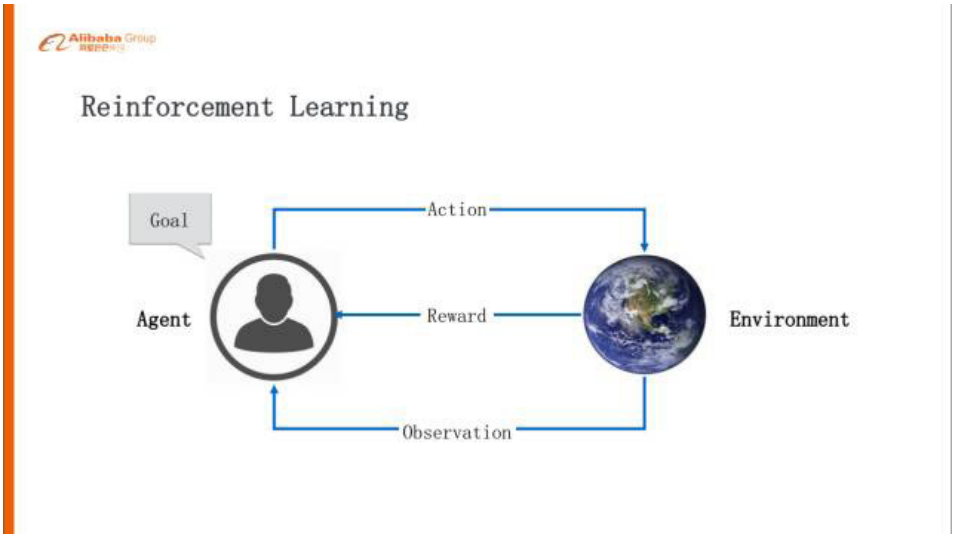
《星际争霸》里面 AI 的研究或者竞赛不是最近才出现的，其实在 2010 年的时候已经有大量的研究人员在研究《星际争霸》里面的 AI，主要是以加拿大 Alberta 大学为主的研究力量，包括一些老师和学生，而且有三个固定的竞赛和一些循环赛，大家在上面 PK。

这一类 AI 的话是 Classic AI，也就是没有学习能力、没有模型、也不需要训练，而是基于预编程的规则，所以不是非常灵活，这种算法下的 AI 其实离真正超过人类或者打败人类目标还是非常非常远的，它们可以打败内置的 AI，但是还远远比不上人类的专业选手，甚至连普通选手基本上也打不过。

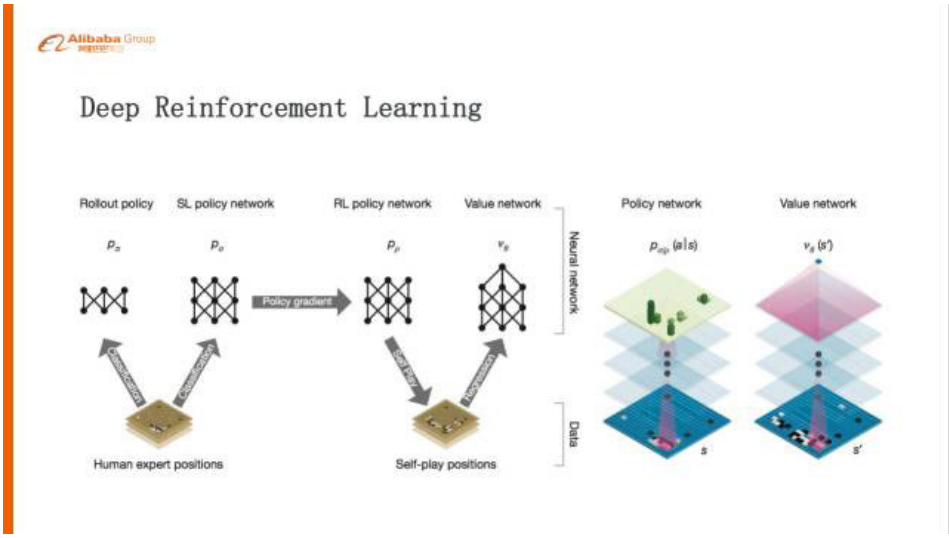
另外一类是 Modern AI，也就是以智能体自主学习为主的算法，从去年开始这

个领域火起来了。比如阿里巴巴和伦敦大学学院，最近在合作的基于《星际争霸 1》里面做一些新的 AI 的尝试。另外就是 Google DeepMind，去年 11 月份他们和暴雪合作，会基于《星际争霸 2》去开放一个 API，让大家基于《星际争霸 2》开发自己的 AI 算法，另外像 Facebook 也有一些团队做这方面的研究。

2. 深度强化学习



强化学习是非常接近人类学习的一个学习机制，通过这个 Agent 跟环境的交互，在交互当中学习。Agent 会观察周围的环境，然后环境会给它一些反馈，Agent 根据状态和反馈会做出一些动作，这些动作会或多或少的影响这个环境，环境会根据这个动作反馈一些 Reward，Reward 可能是奖励的也可能是惩罚的，Agent 根据这样的试错，不断的去调整。

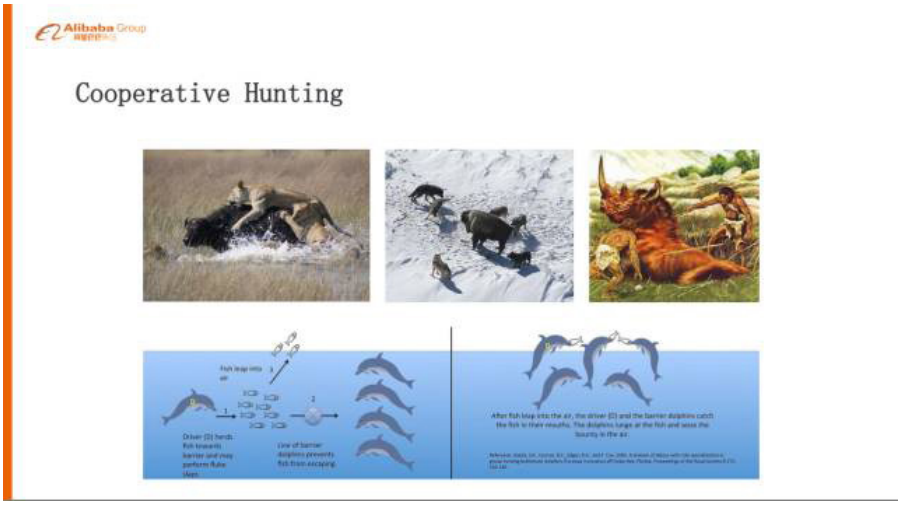


Agent 背后有两个概念非常重要，一个是不停的优化策略，什么样的状况下采用什么样的 Action 是合理的，另外一个是用价值函数评估当前的状态它的价值是怎么样的。

强化学习跟深度学习结合，就叫深度强化学习。因为深度学习或者神经网络是非常适合去做这种表示学习的，可以表示成一个复杂的函数。policy 或者 value 用神经网络去逼近的话，在工程上或者效率上是非常好的提升。

以 AlphaGo 的例子来讲，在训练的时候分成三个阶段，第一个阶段，从人类的棋谱里面学习人类的先验的知识，通过监督学习学习一个较好的、胜率较高的 policy network；第二个阶段，基于监督学习学习出来的 policy network，然后自我对弈，通过 policy gradient 再去优化 policy network，这就比之前学出来的 policy network 要更好；第三阶段，再用学出来的强化学习版的 policy network 自我对弈，获得一个最佳。

3. 多智能体协作



其实目前为止所有的 AI 的智能体比较成功的一些应用基本都是这种单个的 Agent，其实对于人类来讲，协作智能是智能体的一个非常大的方面，我们的祖先智人为什么可以统治地球，其中一个很大的原因就是，他们学会了大规模的协作，而且是非常灵活的协作。可以想象一下，未来全部都是这种 AI 的智能体，它们能不能自我学习到人类水平协作的一个智能呢?

我们用了一个词 Artificial Collective Intelligence，这对现实和未来都有非常大的意义。比如手机淘宝，现在绝大部分流量背后都是一个算法推荐出来的，不管广告还是搜索其背后都是 AI 的智能体在做，目前这些智能体都是各出各的优化，或者推出自己的商品。

Alibaba Group

Multiagent Bidirectionally-Coordinated Nets for Learning to Play StarCraft Combat Games

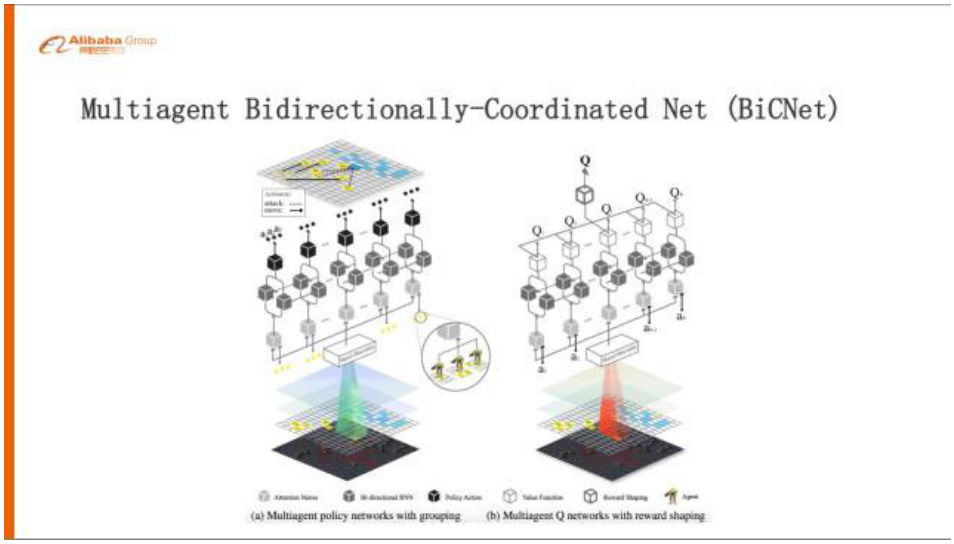
Peng Peng†, Quan Yuan†, Ying Wen‡, Yaodong Yang‡, Zhenkun Tang†, Haitao Long†, Jun Wang‡ *
†Alibaba Group, ‡University College London

Abstract

Real-world artificial intelligence (AI) applications often require multiple agents to work in a collaborative effort. Efficient learning for intra-agent communication and coordination is an indispensable step towards general AI. In this paper, we take StarCraft combat game as the test scenario, where the task is to coordinate multiple agents as a team to defeat their enemies. To maintain a scalable yet effective

https://arxiv.org/abs/1703.10069

其实我们在考虑的是，比如手机淘宝首页里边有爱逛街、猜你喜欢这种位置，那么他们能不能够协同地去推荐一些这样的商品，从而可以让用户的体验最好，让平台的价值最大化。其实以后可能都是算法经济、AI 经济，都是这种 AI 的 Agent，比如满大街可能都是自动驾驶的无人车，他们之间是不是也需要一些协作，让交通出行效率能够达到最大化。



最近我们在《星际争霸》里的微观战斗场景下，提出来一个多智能体双向协作网络，关于这个网络的详细内容大家感兴趣可以下载我们的 paper 看一下，这个工作是我们跟 UCL 一起合作完成的，用来探索或者解决多智能体协作的问题。

这是我们提出来的 BiCNet(Multiagent Bidirectionally-Coordinated Net) 的网络结构，它其实也是比较经典的结构，分成两部分，左边这部分是一个 policy 的网络，就是说从下往上会把《星际争霸》的环境进行一些抽象，包括地图的信息、敌方单位的血量、攻击力，还有我方 unit 的信息，抽象出来形成一个共享的 State，经过一个双向的 RNN 这样的网络，进行充分的双向的沟通，再往上每个 Agent 去得出自己的 Action。

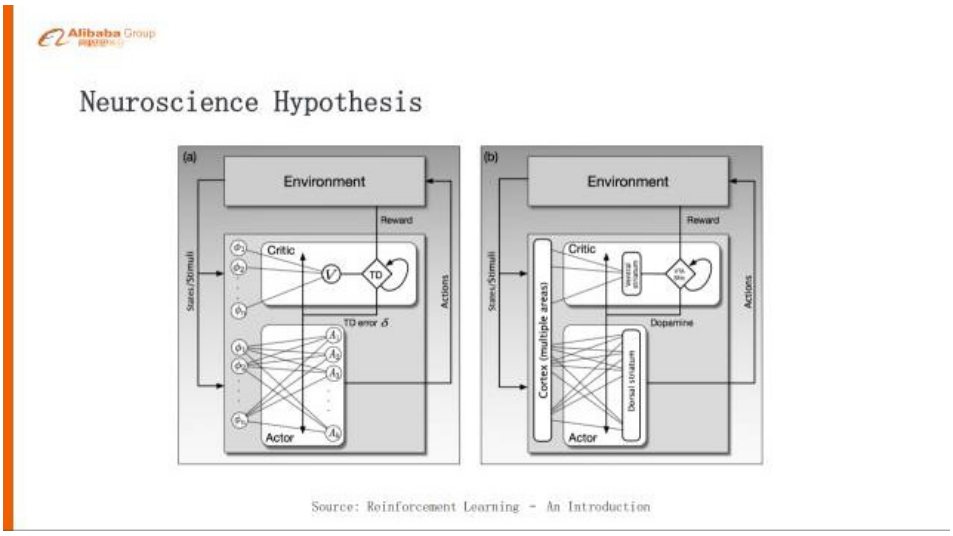
比如我到某一个地方去攻击谁。左边这个 policy network 就是对于当前的状态应该采取什么行动，右边就是一个 value 的 network，根据前面 policy 得出来的 Action，还有抽象出来的 State 进行评估，Q 值大概是多少，做出一个预判。当采取这些行动以后，这个环境就会给出相应的反馈，一些 Reward 来说明这步打的好还是不好，然后会通过一个 Reword 从右边这个网络下来，去反向传播更新里面的参数。

这个网络有几点比较好的设计：

第一，它的 scalability 比较好，《星际争霸》里面打仗的时候随时可能会有伤亡，

这个 Agent 死掉以后这个网络不是还可以正常的工作，包括源源不断涌现的新的 Agent 进来，是不是也是可以工作。我们看到双向网络参数是共享的，所以是不会有影响的；

第二，我们在中间用了这样一个双向网络以后，其实是在一个效率和性能之间做了比较好的平衡，如果用全连接网络的话，计算量会过大。但是我们用一个双向网络，前面告诉你大概要做什么样的 Action，回来再告诉前面的人他们采取了什么样的 Action，一结合，最后算出来应该追加的策略是什么样子，从实际来看效果也是非常好的。

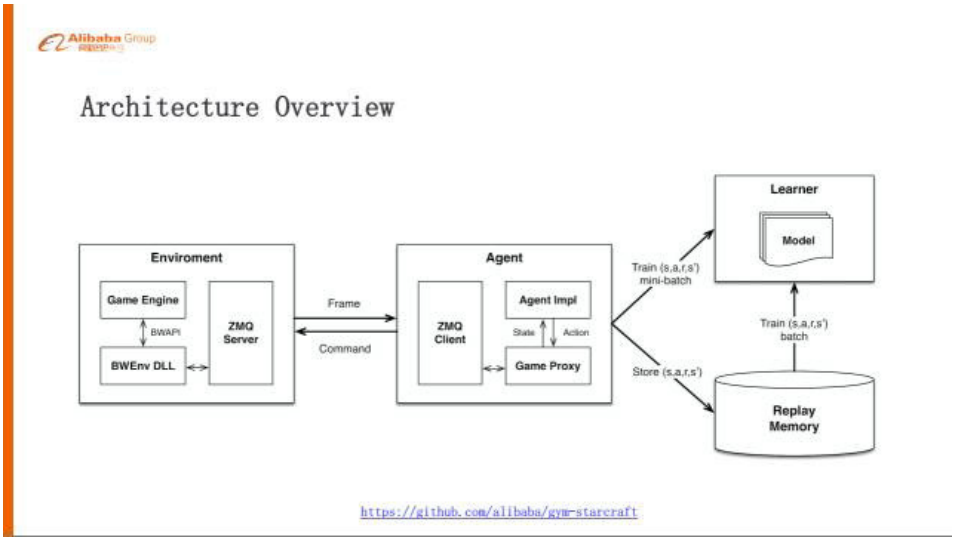


其实我们认知计算实验室在设计一些算法或者模型的时候会参考神经科学里边目前的一些研究成果，我们认为研究认知心理学、大脑、类脑的研究或者神经科学，对于做人工智能应该有两个好处。

第一个好处就是，神经科学具有启发性，就是当你在一些具体的问题或者场景里面去思考的时候，会遇到一些问题，这些问题可能是从来没有人解过的，如果神经科学、交叉科学里有类似的这种结构或者算法，这些可能会很好的解决你的问题，带来算法上的一些启发。反过来另外一点，神经科学也可以帮你做验证，你设计一个算法以后，如果神经科学里面有类似的结构，那么很大概率这个算法是可以工作的。

其实我们的 Actor-Critic 网络在人脑里面也是有相应的对应，左边就是 Actor-Critic 这个网络，右边是我们的大脑，大脑里边纹状体就是负责 Actor、Critic 两部分，这个纹状体腹部是负责 Critic 这部分，背部是负责 Actor 这部分，Reward 下来以后我们大脑会计算，这与预期的 Reward 有什么差距，这个差距就会以多巴胺的形式影响到 Actor，下一次你就要按照这个去调节，让下一次 Action 做的更好一点。其实多巴胺体现在我们的算法里面就是 TD error，也就是我们算的 Reward 的误差，这其实是一个很好的对应。

4. 实验平台和实际效果



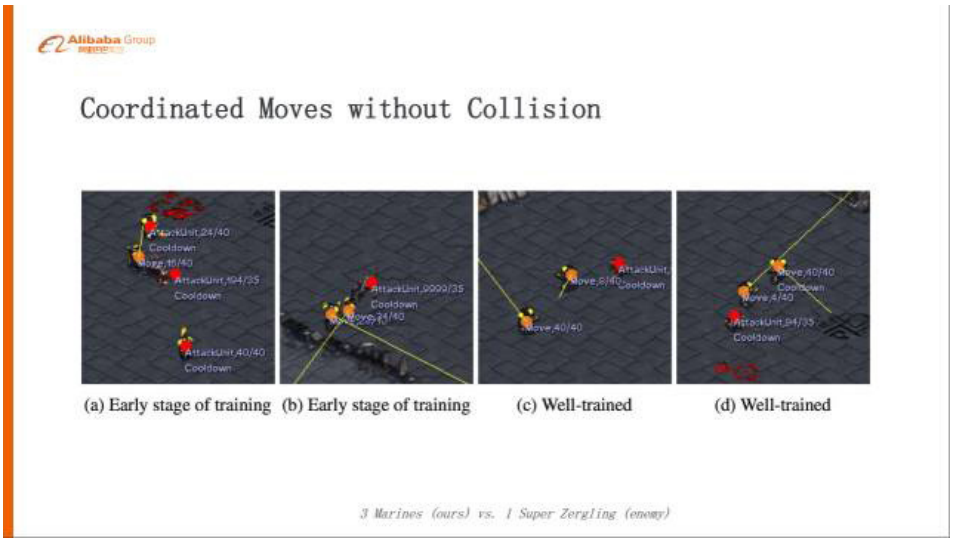
前面是网络架构的设计，为了实现这样一个算法模型，我们搭了一个实验平台，这个实验平台就是基于 Facebook 的 TorchCraft，它是把《星际争霸 1》和 Torch 封装在一起，但是我们比较习惯于 TensorFlow 和 Python，所以在上面做了一个封装，再把这套架构放在这个 OpenAI 标准接口里边，大家有兴趣可以试一下。

这个架构主要分成两部分，对应刚才说的强化学习，左边是 Environment，其实就是《星际争霸》这个游戏，包括引擎，还有里面的 DLL，DLL 基于 BWEnv，这是一个官方认可的 DLL。基于这个 BWEnv DLL 把内部的状态、指令封装起来，

其实这就是一个 Server，右边就是 Agent，是一个 Client，这样你可以连上很多的 Agent 玩这个游戏。

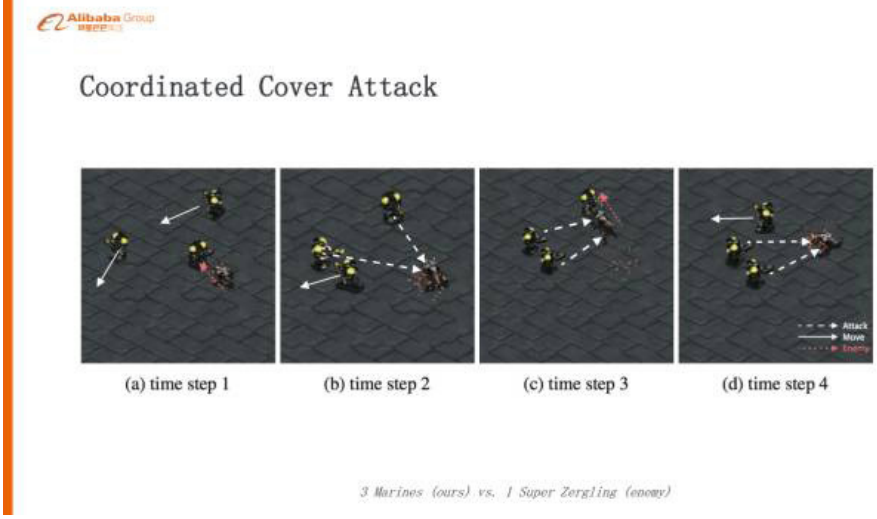
中间是传递的信息，Environment 会把它每一帧的数据吐给 Agent，Agent 会把每一帧的数据抽象成状态，然后再把这个 State 送到 model 里面去学习或者做预测，反过来会预测出来一些 Action，这些 Action 会封装成指令，再发回给《星际争霸》的 Environment，比如说开枪或者逃跑，这个是我们搭的这样一个《星际争霸》的实验平台。

下面是我们这个实验平台做到的一些效果，总结起来有五种可观测的智能：



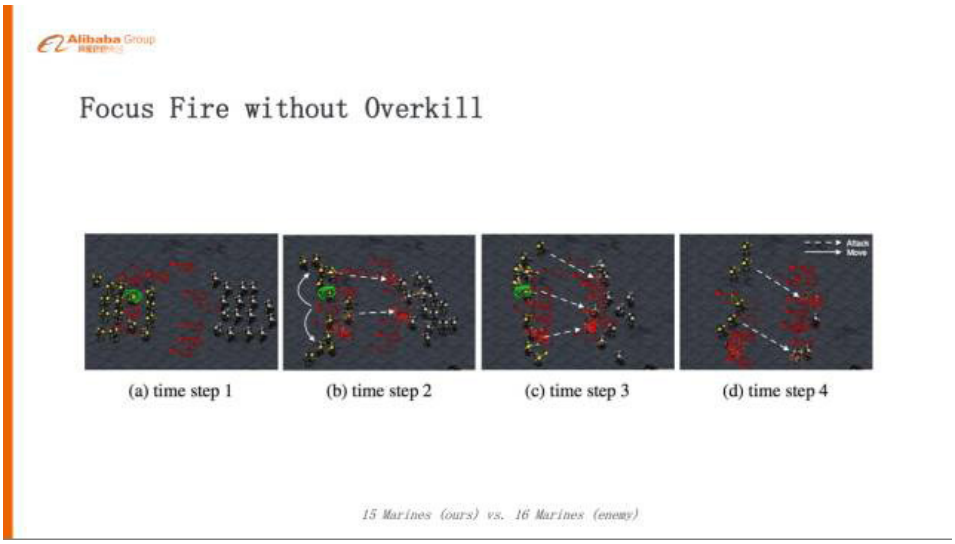
第一种，可以配合走位。这个例子就是三个枪兵打一个 Super 的小狗，这个小狗是我们编辑过的，血量非常大，一下子打不死。三个枪兵打一个小狗，a/b 这两个图，在训练的早期其实是没有学会太多的配合意识，所以他们走位的时候经常会发生碰撞，经过可能几万轮的训练以后，他们慢慢学会了配合队友的走位，这样大家撞不到一起。

第二个场景，这个配合就是边打边撤，Hit and Run 这样的技能，这个例子就是三个枪兵打一个狂徒，利用远程攻击的优势来消灭敌人。

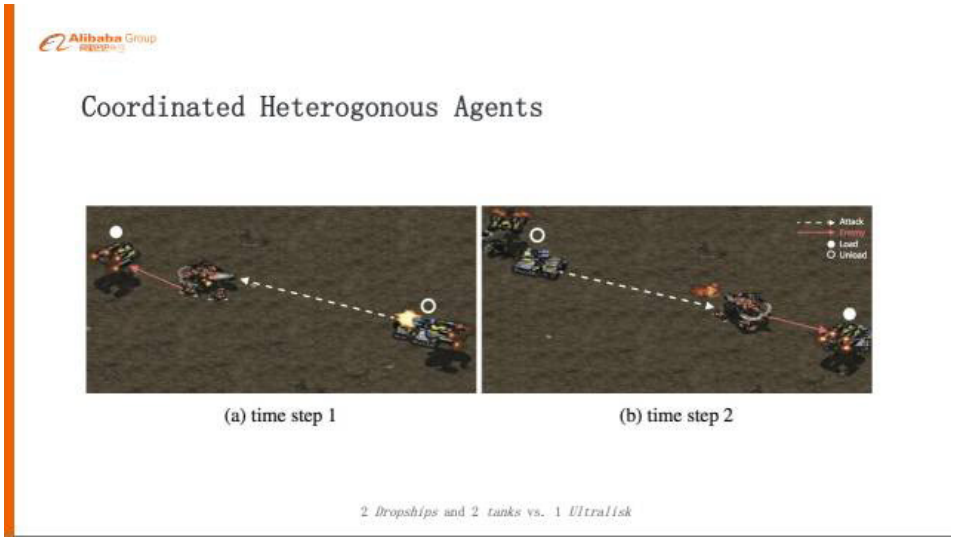


第三种，掩护攻击。刚才三个枪兵打一个狂徒的时候是同时撤退，但是在这个场景下有些枪兵可能会去吸引这个小狗或者去阻挡一下，让另外两个枪兵抓住这个时间

空隙来消灭这个小狗。非常有意思的一点就是，这种协作不是在任何情况下都会出现的，如果你的环境不是那么的有挑战性，可能它就是简单的 Hit and Run 就足够了，如果我们的环境更严苛一点，比如这个小狗血量调高，攻击力从 3 调到 4，或者血量从 210 调到 270，发现它又学会了另一种更高级的掩护攻击的协作，这就非常有意思了。



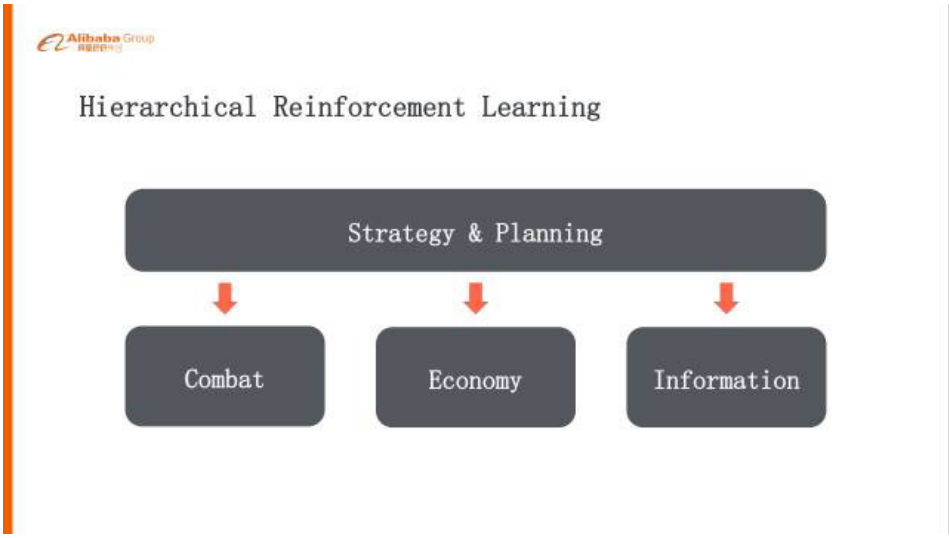
第四种，分组的集火攻击。这个例子是 15 个枪兵打 16 个枪兵，大家想想应该怎么取胜？策略可能 3 个枪兵或者 4 个枪兵自动组成一组，这 3 个枪兵先干掉一个、再干掉一个，就是把火力集中，但又不是 15 个枪兵打 1 个，而把火力分散一点，最后可能我们这方还剩 6 个枪兵，对方可能全部消灭掉了，这个都是通过很多轮次的学习之后他们自动去学到的这样一个配合。



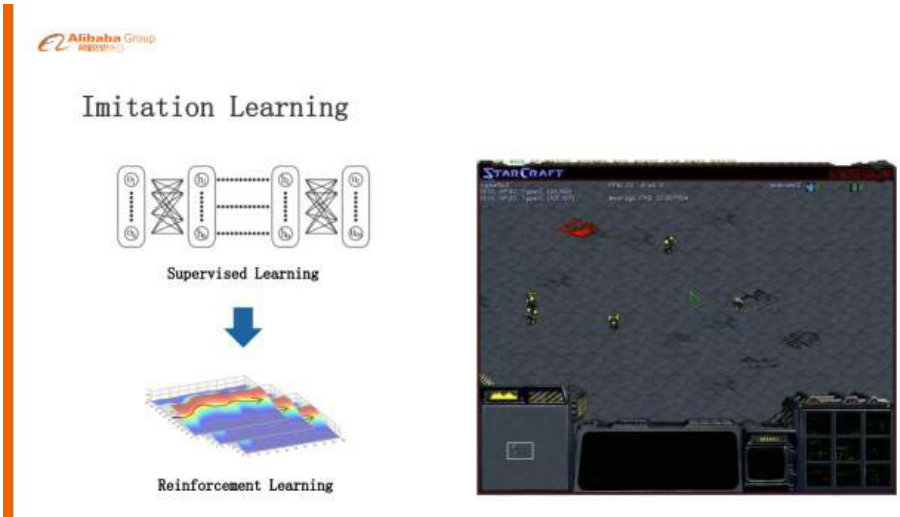
第五种，不光是枪兵之间学会配合，还可以多兵种配合，异构的 Agent 的配合。这个例子就是，两个运输机，每个运输机带一个坦克去打一头大象，正常来讲，两个坦克打一个大象肯定是打不过的，加上运输机的配合以后，大象攻击某一个坦克的时候，运输机会及时的把这个坦克收起来，让大象扑空，同时另外一个运输机赶紧把它的坦克放下去，去攻击大象，这样一来一回可能大象一点便宜占不到就被消灭了，这个是基于我们之前的做出 BiCNet 一个协作的展现。

5. 关于未来的一些思考

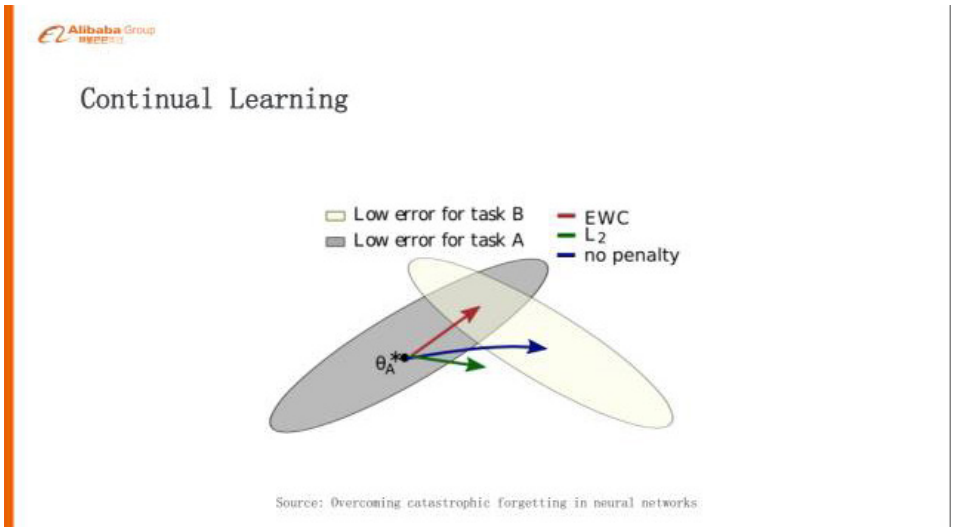
但是《星际争霸》里其实不光是微观战斗，其实更难的是宏观的策略方面，怎么样“宏观 + 微观”打一整个游戏，这样其实我们也有一些思考，可能不是特别成熟，但是我们可以一起探讨一下。



第一点，要玩一个 full-game，如果是简单的单层次的强化学习，可能解决不了问题，因为 action space 实在太大了，一个比较自然的做法就是做层级式的方式，可能上层是策略规划，下面一层就是它的战斗、经济发展、探路、地图的分析等等，这样的话一层一层的，就是高层给下层设置一个 goal，下层再给下面一层设计一个 goal，其实这跟人的问题分解是比较类似的。



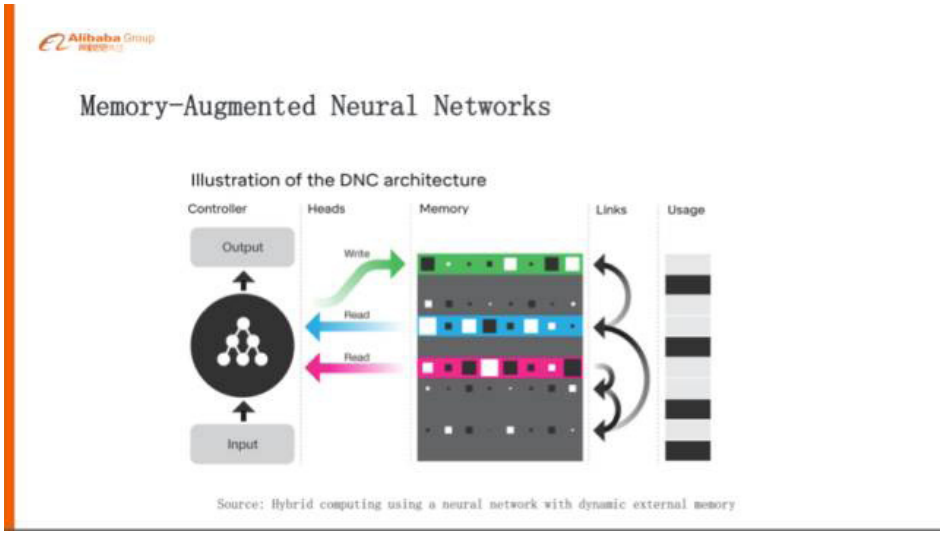
第二点，我们觉得值得去研究和探讨的是模仿学习，Imitation Learning，刚刚讲的 AlphaGo 的例子也是 Imitation Learning，第一步通过监督学习学习比较好的策略，再把监督学习学好的策略通过自我的对弈去提升，在《星际争霸》里面更需要这种模仿学习，比如说我们两个枪兵打一个小狗的时候，我们认为一个好的策略是一个枪兵吸引小狗在那儿绕圈，然后另外一个枪兵就站在中心附近开枪，把这个小狗消灭，两个枪兵一滴血可以不死。但是这种策略是比较难学习的，所以我们先给它人为的让这个枪兵在里面画圈，画上几步之后枪兵自己学会画圈了，带着小狗，然后另外一个枪兵在后面追着屁股打，这种探索就非常的有效。



第三点，我们叫 Continual Learning，如果要迈向通用智能，这是绕不过去的课题。Continual Learning 像人一样，我们学会了走路，下一次我们学会了说话，我们在学说话的时候可能就不会把走路这件事情这个本领忘掉，但是在《星际争霸》一些场景的时候，神经网络学到 A 的时候再去学 B，这个时候可能会把 A 的事情忘掉。

举个例子，一开始我们训练一个枪兵打一个小狗，这个小狗是电脑里边自带的 AI，比较弱，这个枪兵学会了边打边撤，肯定能把小狗打死。我们再反过来训练一个小狗，这个小狗去打电脑枪兵，这个小狗学会最佳策略就是说一直追着咬，永远不要犹豫，犹豫就会被消灭掉，所以它是一条恶狗，一直追着枪兵咬。

然后我们把这枪兵和小狗同时训练，让他们同时对弈，这样发现一个平衡态，就是枪兵一直逃，狗一直追，《星际争霸》设计比较好的就是非常平衡。然后这个枪兵就学会了一直跑，我们再把这个枪兵放回到原来的环境，就是再打一个电脑带的小狗，发现它也会一直跑，它不会边打边撤。你发现它学习的时候，学会了 A 再学会 B，A 忘了，这个其实是对通用人工智能是非常大的挑战，最近 DeepMind 也发了一个相关工作的 Paper，这也是一个 promising 的方向，大家有兴趣可以去看一下，他们的算法叫 EWC。



最后一点，前面有说到几大挑战，其中有一大挑战就是长期的规划，长期规划里边我们认为一个比较好的做法就是，给这种强化学习里面去引入 Memory 的机制，这也是目前的一个比较火的方向，像 Memory Networks、DNC，要解决的问题就是，我们在学习的过程当中应该记住什么东西，从而使得我们可以达到一个很好的最大的 Reward。

所以今天跟大家交流的主要就是说，其实在《星际争霸》里面是蕴含了非常非常丰富的研究通用人工智能或者研究认知智能的场景，这个里面可以有很多非常有意思的课题。我只是列举了四个方向，其实还有很多很多方向可以去研究。欢迎有兴趣的同学跟我们一起来认真的玩游戏。谢谢大家！

学术前沿

KDD 论文解读 | 想要双 11 抢单快？靠这个技术提速 9MS

席奈

6 月 29 日，阿里巴巴在杭州召开 2017 天猫双十一发布会，宣布启动：双 11 超级 IP 计划。今年晚会将由北京卫视、浙江卫视、深圳卫视三台同时直播。淘宝直播、优酷等在内的多家平台同步跟上，让澳门、香港、新加坡等地也能同步收看天猫双 11 晚会，相信今年的双 11 一定会成为举世瞩目的全球狂欢节。

同时，为 2016 双 11 提供技术支持的团队也首次曝光了其研究成果，通过 CLOSE 排序算法，2016 双 11CPU 的使用率降低了约 45%，搜索的平均延迟下降了约 30%（平均的搜索 latency 从 33ms 下降到 24ms），同时 CLOES 本身带来的 GMV 提升了近 1%。相关论文《Cascade Ranking for Operational E-commerce Search》也被国际数据挖掘顶会 KDD 2017 收录。

该论文设计并实现了一种级联式电商搜索方式：它的主要思想是将一次排序分成递进的多个阶段，各阶段使用逐渐复杂的特征去得到逐渐准确的结果。在靠前阶段使用简单特征过滤显然不合要求的结果，在靠后阶段使用复杂特征辨别难以区分的结果。除此以外，算法结合电商场景的特殊性，严格限制了引擎的响应时间以及返回商品的数量，以保证用户的搜索体验。

离线实验和在线实验均验证了算法的正确性以及有效性，对比传统的方法能提升准确率的同时大幅提升了计算性能；在去年双 11，在新增了大量准确又耗时的计算特征（包括强化学习和深度学习特征）的情况下，算法极大的保证了引擎的效率，使排序对引擎的压力下降 40%，同时使排序效果有较大提升。

针对这篇论文，阿里妹公布独家技术解读：

面临的问题

淘宝的搜索系统无疑是全球最大的电商搜索系统。“最大”这里包括商品量、用户量，包括引导的成交额、点击成交量，还包括引擎的访问次数、访问 QPS…这样一个搜索引擎，所需要面对的访问压力也是巨大的，尤其在“双十一”等大促场景，压力更是平时的数倍。

另外一般搜索引擎的目标主要是引导点击，而在电商中，排序的结果更希望引导的是成交量和成交额。因此我们的搜索系统、排序方案需要考虑多种实际问题。首先是在有限计算资源情况下，如何拿到更好的排序结果；其次是怎样保证用户的搜索体验，包括结果返回时间、返回商品量等；最后是怎么保证电商场景下的多目标，包括点击、成交量和成交额。

已有方法的不足

学术界和工业界都有大量 learningto rank 方面的研究，均期望能通过机器学习，为用户给出更优的排序结果。然而绝大部分相关工作都集中在如何提升排序的质量，却并不关系排序的效率，而太低效的排序方案在实际的工业在线应用中，往往是不可接受的。

淘宝搜索和其他类似应用主要采取的解决方案是使用一种“两轮排序方案”：在第一轮使用非常简单的特征去得到一个小的候选集；第二轮在小的集合上做复杂的排序。可是这种启发式的方案并不能保证性能与效果的取舍是最优的。基于以上考虑，我们需要一种全新的、工业可用的、能更合理平衡效率与性能的排序方案。

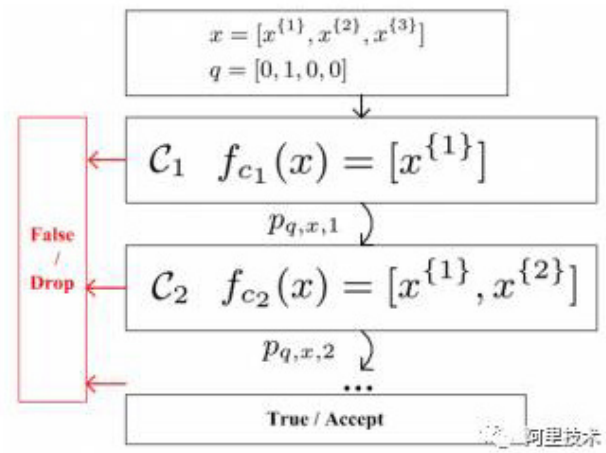
CLOSE 排序算法，平衡性能与效率，保障用户体验

论文受图像中快速目标检测算法的启发，发现并不是引擎中的每个商品都需要全部特征参与计算、排序——一些基本特征能帮助过滤掉大多数商品；逐渐复杂的特征过滤逐渐难以区分好坏的商品；全部特征排序剩余商品。基于这样的思想，论文提出了一种多轮级联排序方法 Cascade model in a Large-scale OperationalEcommerce Search application（CLOES）。

CLOES 主要采用了一种基于概率的 cascade learning 方法，将排序分为多轮计算；将排序效果和 CPU 的计算量作为优化目标，一起建立数学模型，同时优化。除了考虑性能与效率，算法还考虑了用户的搜索体验，保证用户在输入任何一个 query 后都能在限制时间内得到足够的返回结果。最后 CLOES 还考虑了电商场景的特殊性，保障了多目标的平衡与可调整。

平衡性能与效率的排序（Query-Dependent Trade Off Between Effectiveness and Efficiency）

论文最重要的部分就是怎么样去平衡一个排序算法的性能和效率，那么我们主要使用的方法是 cascade learning，即将一次排序拆分成多个递进阶段 (stage)，每个阶段选用逐渐复杂的特征去过滤一次商品集合。同时我们使用 learning to rank 设定，将排序问题转化为一个二分类问题，预估每个商品的点击率。



如图所示，我们记一个商品 x(表示为一个 k 维向量) 在 Query q 下，能通过第 j 个 stage 的概率为，其中表示 sigmoid 函数。那么一个商品最终能被点击的概率为能通过所有 stage 的概率之积：

$$p(y = 1|q, x) = p_{q,x} = \prod_{j=1}^{T} p_{q,x,j}$$

我们通过极大似然估计去拟合样本，使用负的 log 似然来表示损失函数，那么基础的损失函数可以表示为 L_1。L_1 关注的是排序的准确性：

$$L_1(\mathrm{w}) = -\left[\sum_{i=1}^{N} y_i \log(p_i) + (1 - y_i)\log(1 - p_i)\right] + \alpha||w||_2$$

其中左边项表示似然函数，影响模型的准确度；右边项$\alpha||w||_2$表示正则项，一方面是防止过拟合，另一方面能预防特征相关导致的 ill-condition 问题。

由于在实际的搜索排序中，我们除了效果，性能也是不得不关注的部分，因此我们需要将系统的性能性能消耗也加到目标中。我们可以求 CPU 的总消耗等于每个 stage 下的性能消耗之和：$\tau(\mathrm{w}) = \sum_{j=0}^{T} \mathbb{E}[Count_j] * t_{j+1}$。其中$\mathbb{E}[Count_j] = \sum_{i=1}^{N} p_{qi,xi,pass_j}$表示每个 stage 上需要计算的商品量的期望，$p_{qi,xi,pass_k} = \prod_{j=1}^{k} p_{q,x,j}$表示商品 x 能进入第 j 个 stage 的概率，t_j表示在第 j 个 stage 上的 feature 进行一次计算的总耗时。那么我们得到一个新的 loss L_2，L_2除了考虑排序的效果，兼顾了模型的计算量：

$$L_2(\mathrm{w}) = -\mathrm{l}(\mathrm{w}) + \alpha||w||_2 + \beta\tau(\mathrm{w})$$

通过调整β，我们能调节系统的性能与效率。β越大，系统负载越低，但排序结果也越差；β越小，排序结果越好，但系统开销越大。

用户体验保障（Multiple Factors of User Experience）

如果直接使用上述模型，确实可以直接降低引擎的负载，但是仍然存在 2 点用户体验上的问题：一是对于某些 query（特别是 hot query），可能计算 latency 仍然会非常高；二是某些 query（一般是长尾 query）下，返回给用户的结果特别少。那么为了解决这 2 个问题，我们进一步的增加了 2 个约束：单 query 下的 latency 不能超过 100（只是举例，不一定是 100）ms；返回给用户的结果数不能小于 200。那么很自然的，我们会想到使用类似 SVM 的 loss 形式：

$$\hat{w} = \arg\inf_{w} \sum_{i=1}^{N} \xi_i + L_2(\mathrm{w}) \quad s.t.\ \ Count_{q_i,T} \geq N - \xi_i$$

上述公式可以比较直观的理解为当 query 下的 latency 小于 100ms(N 的值)的时候，loss 为 0；大于 100ms 时，loss 为（latency-100）的线性倍数；返回结果数类似。然而该函数是非凸、不可导的，并不利于问题的求解。因此为了求解的方便，我们使用了一个凸近似函数 modified logistic loss 去逼近 SVM loss，可以证明，该 loss 和 hinge loss 是几乎一致的，当我们取一个较大的γ的时候：

$$g'(z,N) = \frac{1}{\gamma}\ln\left(1+\exp\left(\gamma(N-z)\right)\right)$$

综上，我们考虑了用户的 2 种体验之后，最终的目标函数可以写成下面形式：

$$L_3(\mathrm{w}) = -\mathrm{l}(\mathrm{w}) + \alpha||w||_2 + \beta\tau(\mathrm{w})$$

$$+\delta\sum_{i=1}^{N} g'\left(Count_{q_i,T}, N_o\right) + \epsilon\sum_{i=1}^{N} g'\left(T_l, Latency_{q_i,T}\right)$$

其中N_o表示期望返回给用户的最少结果数（例如 200），T_l表示希望的最大 latency（例如 100ms）。通过最小化L_3，我们既能在有限的计算资源下得到更好的排序结果，又能兼顾用户的搜索体验。

商品场景下的多目标（Importance Factors of E-commerce Search）

电商搜索与网页搜索或者广告有较大区别：我们关注的不仅是点击 ，成交量、成交额等指标同样重要。然而如果我们将所有正样本(点击和成交)一样处理，由于点击样本量远大于成交样本，那么我们更像在学习一个 CTR 任务；这在我们想得到更高的成交额或 GMV 时是不合理的。因此我们为不同类型、不同价格的正样本设置了不同的权重。更具体的，我们会区分样本商品的 log（价格）、点击和成交，于是在表示准确的似然项上，做了如下修正：

$$\mathrm{l}(\mathrm{w}) = -\left[\sum_{i=1}^{N} y_i \log(p_i) + (1-y_i)\log(1-p_i)\right] wgt_i$$

$$wgt_i \begin{cases} \varepsilon * \mu * \log(price) & if\ x_i \text{ 是成交样本} \\ \mu * \log(price) & if\ x_i \text{ 是点击样本} \\ 1 & if\ x_i \text{ 是负样本} \end{cases}$$

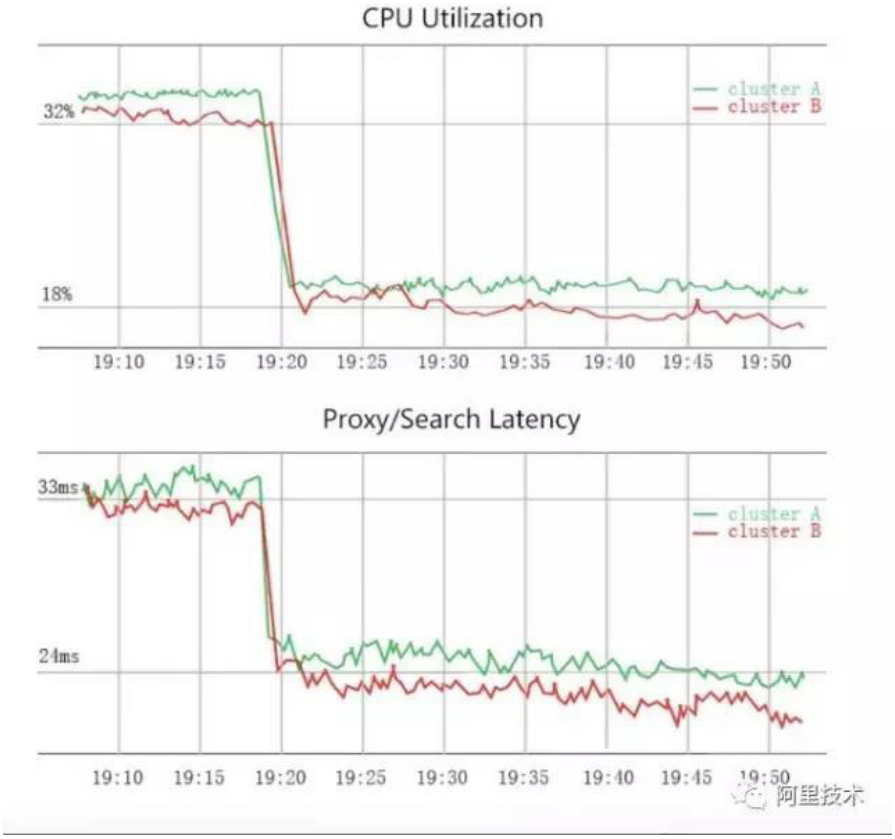
在上式中，ε越大，成交样本的权重更高；μ越大，价格因素的影响越大。权重的作用主要会体现在优化过程的梯度求解上。

离线与在线验证

为了验证算法的有效性，我们随机采样了线上一天的日志做交叉验证，数据取自 2016 年 10 月底。我们主要考察的指标有 2 点：测试集上的 AUC 以及性能总消耗。对比的算法有 1，使用全部特征做一次排序；2，使用简单特征做一次排序；3，线上使用的 2-stage 方法；4，CLOES 算法，取β = 1；5，CLOES 算法，取β = 10。实验结果如下表。从表中我可以看到使用全部特征的准确率无疑是最高的，然后计算消耗也是最高的；线上使用的 2-stage 方法能显著的降低计算效率的问题，只有方法 1 的 30%，但是 AUC 也降低到 0.76。我们主要对比的是现在线上使用的方法 3—2-stage approach，使用了 CLOES，在几乎相同的计算消耗下，AUC 能从 0.76 提升到 0.80；在几乎相同的 AUC 下，计算消耗能从 30%进一步下降到 18%。

	Train set AUC	Test Set AUC	Cost
Single stage with all features	0.88	0.87	1
Single stage with simple features	0.73	0.72	0.07
Online 2-stage approach	0.78	0.76	0.30
CLOES(β = 1)	0.81	0.80	0.29
CLOES(β = 10)	0.80	0.77	0.18

在离线验证了算法效果后，我们在双 11 前夕对算法进行了上线，以期望降低引擎的计算压力。上线期间的引擎 CPU 使用率以及平均搜索 latency 变化如下图：可以看到 CPU 使用率从 32% 下降到 18%；而平均的搜索 latency 从 33ms 下降到 24ms，图中有 2 条曲线分别表示引擎的 2 个集群。需要注意的是，在引擎压力大量下降的情况下，线上的排序指标，包括 CTR 和 GMV 是略上升的。



受益于 CLOES，在双 11 当天，引擎的负载在最高峰也没有超过 70%，CPU 的使用率降低了约 45%，搜索的平均延迟下降了约 30%，同时 CLOES 本身带来的 GMV 提升了近 1%。考虑到其他因为性能改善而能上线的特征（包括实时特征和 RNN 特征等），排序的 CTR 提升有 10%–20%，同时成交量、GMV 等指标也有大幅提升（指标对比基于标准 A/B Test）。

总结

搜索对于电商来说是最大的流量入口，搜索排序的质量对用户的体验、对商家的收入、对平台的效率都会起到至关重要的作用。未来淘宝搜索会继续以用户的搜索体验为主要目标，为用提供能更优质、更能满足用户个性需求的排序结果。

KDD 论文核心算法独家解读

探微

阿里妹提示：全文三千字，内含独家披露的算法公式，阅读预计需要 5-8 分钟。文末有福利。

近日，来自阿里妈妈精准技术团队的论文《淘宝展示广告中的 OCPC 智能调价算法》被国际数据挖掘领域顶级会议 KDD（Knowledge Discovery and Data Mining）收录。

该论文围绕 OCPC 智能调价，创新地提出了一种双层优化形式，将优化广告主价值转化为首要约束条件，将系统根据预估收益对广告进行排序作为内层优化问题，将用户体验和平台收益的最大化作为外层的寻优问题，并提出了相应的求解方法。最终达到不仅广告主效果优化，平台商业收入和用户指标也获得优化的三位一体目标。

研究问题：传统广告系统本质是粗粒度的流量区分和匹配

基于固定出价的传统广告系统中，广告主对特定用户人群和广告位设定固定的竞价，其本质是粗粒度的流量区分和匹配。事实上，广告主有着进一步细粒度的竞价和流量质量匹配的诉求。

该系统还有着两方面的缺陷。其一，广告主单一固定的出价应对连续多变的流量模式导致经济效率低下；其二，传统的最大化广告收益（eCPM）的排序机制过于追求短期商业利益，无法调控流量对用户体验、商家利益、成交额等指标的影响，不利于淘宝生态的长期可持续繁荣。

研究场景：最大在线市场淘宝网在线广告系统

曾被《经济学人》称为“中国最大在线市场”的淘宝拥有世界先进水平的在线广告系统。该论文的研究聚焦于重要的淘宝移动端 CPC 展示广告中的竞价优化问题。由于淘宝 CPC 展示广告处于相对重要的场景，其广告投放的优化通常需考虑诸多因素，如反应商家利益的投资回报率（ROI）、总成交额（GMV）、转化率（CVR），反应

平台用户体验的点击率（CTR），以及淘宝平台收益指标（千次展现收益 RPM）等。

值得一提的是，谷歌公司 AdWords 的 ECPC 也尝试根据用户潜在转化率调整广告主竞价。然而，除了转化率，ECPC 无法直接优化其它诸多对淘宝生态重要的指标（如平台整体 GMV 等）。

研究设想：OCPC 智能调价算法实现自动调整竞价

论文设想通过算法对于每条用户请求，在保障优化广告主利益的前提下，自动地调整广告主竞价从而使竞价能够反应该流量的真实价值。流量价值的定义上，算法融合了用户体验、广告主收益以及平台收益的整体生态指标，旨在实现三方共赢的商业局面。基于此，论文提出了一种新的 Optimized cost per click (OCPC) 智能调价算法。

OCPC 智能调价算法从数学上描述并分析了优化广告主诉求的条件，进而提出了一种优化平台生态综合指标以及平台收益的算法。事实上，该算法框架适用于多种广告主诉求以及平台生态指标的优化问题，例如用户的浏览量、点击量、转化率等。

论文选择了 ROI 以及流量质量作为广告主的优化诉求，选择 GMV 作为平台生态指标，并通过调整广告主出价优化平台的商业收益。

论文核心算法

ROI（投资回报率）优化算法

智能调价算法分析了优化广告主投资回报率的条件，即 ROI 约束。其中，定义交易转化为 c，用户为 u，广告位 a，那么在用户和广告的条件下产生转化的条件概率为 p(c|u, a)。对于一个特定的广告计划 a，定义为预估的笔单价。因此，单次点击的期望 GMV 为。进一步定义广告主为每一次点击付费为，那么单次点击的期望 ROI 则为公式（1）：

$$roi_{(u,a)} = \frac{p(c|u,a) * v_a}{b_a} \quad (1)$$

即单次点击的期望投资汇报率为期望 GMV 除以单次点击价格。进而，广告 a 对

于不同用户和点击的 ROI 如公式（2），即平均转化率乘以比单价除以单次点击价格，其中是某个用户在一段时间内的点击量。

$$roi_a = \frac{v_a \cdot \sum_u n_u \cdot p(c|u,a)}{b_a \cdot \sum_u n_u} = \frac{E_u[p(c|u,a)] \cdot v_a}{b_a} \quad (2)$$

公式（2）说明了 ROI 和一段时间内平均转化率 $E_u[p(c|u,a)]$ 之间的线性关系，因此，只需要调价幅度小于当前流量转化率和历史平均流量转化率的比值，就能从理论上保证 ROI 不会下降，如公式（3）所示：

$$\frac{b_a^*}{b_a} \leq \frac{p(c|u,a)}{E_u[p(c|u,a)]} \quad (3)$$

从而，该算法竞价优化的原则是：对于转化率升高的流量，提升其竞价来帮助广告主竞得优质流量；对于转化率下降的流量，降低竞价从而节约在劣质流量上的成本。 出于调价的安全考虑，该算法设定了固定的调价范围参数，保证广告主的调价不过高或者过低，并以此得出调价的上界和下界如公式（4）。

$$l(b_a^*) = \begin{cases} b_a \cdot (1 - r_a), & \frac{p(c|u,a)}{E_u[p(c|u,a)]} < 1 \\ b_a, & \frac{p(c|u,a)}{E_u[p(c|u,a)]} \geq 1 \end{cases}$$

$$u(b_a^*) = \begin{cases} b_a, & \frac{p(c|u,a)}{E_u[p(c|u,a)]} < 1 \\ b_a \cdot \min(1 + r_a, \frac{p(c|u,a)}{E_u[p(c|u,a)]}), & \frac{p(c|u,a)}{E_u[p(c|u,a)]} \geq 1 \end{cases} \quad (4)$$

综合指标优化算法

在给定上下界的范围内优化出价可以帮助广告主获得更高质量的流量以及更高的 ROI。但是，约束空间内解的不同出价会导致不同的 eCPM 排序，并最终影响平台

收益和其它指标的效果。因此，该算法提出了一种在 eCPM 排序机制下，对综合指标保优的出价调整算法。其核心解决的优化问题如公式（5）。

$$\max_{b_1^*,\cdots,b_n^*} \quad f(b_k^*)$$
$$s.t. \quad k = \operatorname*{argmax}_i \ pctr_i * b_i^* \qquad (5)$$
$$l(b_i^*) \le b_i^* \le u(b_i^*), i = 1, \cdots, n$$

其中 n 是候选广告的数量，pctr 是预估点击率（由阿里妈妈精准技术团队自主研发的混合逻辑回归算法 MLR 支持）。f() 函数定义了系统期望最大化的综合指标，典型的两个例子如公式（6），其中，第一个目标函数可以优化总成交额 GMV；第二个目标函数权衡优化总成交额 GMV 和广告收益。

$$f_1(b_k^*) = pctr_k * pcvr_k * v_k,$$
$$f_2(b_k^*) = pctr_k * pcvr_k * v_k + \alpha * pctr_k * b_k^* \qquad (6)$$

如公式（5）所示，我们希望在满足广告主诉求的调价范围内进行调价，使得 ECPM 排序机制（如公式（5））下最优的广告，其综合指标函数 f() 的值也最大。 该优化问题的主要求解方法请参见 KDD 2017 收录的论文原文：Han Zhu, Junqi Jin, Chang Tan, Fei Pan, Yifan Zeng, Han Li, Kun Gai. Optimized cost per click in Taobao display advertising. ACM SIGKDD 2017.

该优化问题的设计实现了最终排序指标和广告主流量目标的解耦。一方面，候选广告仍然按照最大化 eCPM 的 pctr*b 标准排序，广告主通过调价匹配流量价值，同时整体 eCPM 排序机制保证了广告主出价撬动流量的能力不受影响；另一方面，广告平台可以通过设计不同的 f() 函数优化相应的平台综合指标。

双重验证：离线模拟实验和线上实际环境

OCPC 智能调价算法在离线模拟实验和线上实际生产环境中都取得了明显的效

果提升。单品广告离线模拟效果如表 1 所示。其中，评价指标为 RPM（广告千次展现收益），GPM（千次展现 GMV），CTR（广告点击率），CVR（成交转化率），PPC（单次点击扣费）。

基准策略是传统的广告排序方法，表中数据为该论文提出的 OCPC 算法相对基准策略的效果提升百分比。实验结果表明，OCPC 算法明显提升了商家千次展现成交额（GPM）和成交转化率（CVR）。

表1　单品广告离线模拟实验效果提升比例

	RPM	GPM	CTR	CVR	PPC
% Improved	5.6%	14.1%	-1.9%	14.9%	9.5%

OCPC 算法在在线生产环境中也展现了明显的效果提升，相比基准策略，其效果提升百分比如表 2。

表2　单品广告线上生产环境效果提升比例

	RPM	GPM	CTR	CVR
% Improved	6.6%	8.9%	-1.3%	5.2%

持续一周的线上生产环境效果分析表明，OCPC 使得 67% 的广告主制定的广告计划（Campaign）的千次展现成交额（GPM）和广告投资回报率（ROI）同时获得了提升，如表 3。另有 24% 的广告计划显示出流量数量和质量的置换现象（Quantity and quality exchange），其含义是：这类广告主由于获得了更多的流量，拉低整体平均 ROI 下跌。但是，由于其整体流量增幅高于平均 ROI 跌幅，这些广告主的成交转化数都获得了提升。

表3　单品广告线上生产环境广告主优化比例

	% Campaigns
GPM and ROI are improved	67%
Quantity and quality exchange	24%

除了单品广告场景，该算法在淘宝移动端首页顶部的 Banner 广告中也明显提升了商家的成交额和转化率，如表 4。从商品的类目视角分析，表 5 的数据表明 17% 的类目（占 62% 的浏览量）流量的商家 GPM 和 ROI 获得了同时提升，27% 的类目（占 21% 的浏览量）流量的 GPM 获得了提升，另有 30% 的类目（占 12% 的浏览量）显示出流量数量和质量的置换（与单品广告分析相似，其转化效果也获得了提升）。

表4　Banner广告线上生产环境效果提升比例

	RPM	GPM	CTR	CVR
% Improved	3.6%	15.7%	-0.6%	19%

表5　Banner广告线上生产环境广告主优化比例

	% Category	% PV
GPM and ROI are improved	17%	62%
GPM is improved	27%	21%
Quantity and quality exchange	30%	12%

论文指出，持续了一周以上的线上实验结果表明，OCPC 算法在长期的考察中展现了稳定的明显正向效果提升，为大多数广告主以及平台带了经济效益的增长。更多详尽的算法分析以及更多场景中 OCPC 效果的实验请参阅论文原文。

该项技术在淘宝展示广告主要场景上得到了全面应用并取得了明显的效果提升，单品主流场景中 67% 的广告主 ROI（广告投资回报率）和 GMV（商品交易总额）同时获得了提升，另有 24% 的广告主其流量增幅大于 ROI 降幅，即成交总量获得了提升。

该场景商家整体成交额 GMV 提升 8.9%，CVR（成交转化率）提升 5.2%，RPM（广告千次展现收益）提升 6.6%。Banner CPC 场景中，62% 流量的商家成交额 GMV 和 ROI 同时获得提升，21% 的流量 GMV 获得提升，另 12% 的流量成交总量获得了提升。该场景商家总成交额 GMV 提升 15.7%，CVR 提升 19%，RPM 提升 3.6%。

下一步融入人工智能算法

未来，阿里妈妈精准技术团队将继续围绕客户核心价值，在业务上提升营销效率，设计更精准的流量实时价值评估系统。通过持续优化商家和消费者的匹配与连接效率，提升商家的营收和消费者的用户体验。技术上，该团队将进一步探索基于深度学习、强化学习等最前沿的人工智能算法，在持续支持业务效果提升的同时，为学术界、工业界贡献创新的算法和技术解决方案。

论文原文链接：https://arxiv.org/abs/1703.02091

阿里 AI 技术取得重大突破：连破中、英语言处理两项世界纪录

阿里技术

日前，阿里巴巴披露了自然语言处理技术取得的两项新成绩：在全球顶级的知识库构建测评 KBP2017 中，斩获英文实体发现测评全球冠军；在中文语法错误自动诊断大赛（Chinese Grammatical Error Diagnosis，以下简称 CGED）三个 level 中全面夺得冠军，核心指标比其他参赛机构高出一倍。

iDST 自然语言处理首席科学家司罗

司罗是全球权威机器智能学者，曾担任美国普渡大学计算机系终身教授，先后获得美国国家科学基金会成就奖、雅虎、谷歌研究奖等。在阿里巴巴，司罗领导了 iDST 自然语言处理团队，他们的使命是支持阿里大生态（新零售、金融、物流、娱乐、旅行等）的自然语言处理需求；通过阿里云技术输出，赋能广大全网合作者，创造更多商业机会；沉淀技术，和学界、工业界合作者一起创新自然语言技术。

比赛中使用的分词、词性标注和句法分析等基础 NLP 工具都是由该团队自主研发的 AliNLP 平台。该平台支持阿里大生态（新零售、金融、物流、娱乐、旅行等）

的每天多达 600 亿次的自然语言处理需求。团队横跨中国（杭州，北京）和美国（硅谷，西雅图），普遍拥有 10 年以上自然语言处理研发经验，30% 以上有博士学历（如 CMU，伯克利，普林斯顿，清华，北大等）。 团队多次在国际自然语言技术竞赛中取得冠军成绩。

本次 KBP 比赛团队主力 Zhang Qiong, Zhao HuaSha, Yang Yi 等；2017 CGED 比赛团队主力李林琳，谢朋峻，杨毅等。

阿里巴巴夺实体发现测评全球第一

两场比赛中，KBP 是由 NIST（National Institute of Standards and Technology，美国国家标准与技术研究院）指导、美国国防部协办的赛事，主要任务为从自然书写的非结构化文本中抽取实体，以及实体之间的关系。这次测评吸引了全

球 20 多支顶尖团队参与，包括 IBM Research, BBN, Stanford Univ, CMU Univ, UIUC Univ, Columbia Univ, 腾讯等。

这项测评要求 AI 算法在“读完”一篇英文文章后，构建一个物理世界的命名实体和实体之间关系的知识库，如“克林顿和希拉里之间是夫妻关系”、“克林顿毕业于耶鲁法学院”这样一个个实体的关系。

司罗介绍，阿里的算法可以做到对文章上下文的理解。比如，文章出现了 Apple，再出现 Jobs，就可以辨别出这个 Jobs 指的是乔布斯，而不是工作。再比如，文章出现了 Microsoft，那么 Apple 就更有可能是苹果公司，而不是一种水果。

“另外，我们构建了一个算法去学习不同领域之间共同的部分，通过迁移学习提升我们学习的准确度。对于不同领域数据，我们取其精华，去其糟粕，进行智能学习”，司罗说。

在这次测评中，iDST 团队采用经过改良的深度神经网络架构对文本进行理解。改良的架构有三个主要特点：首先该模型可以自动阅读海量文章（如维基百科）并从中汲取经验；其次，该架构可以智能选择训练数据集以保证训练数据的准确性；最后，采用 post regularization 的办法保证模型结果的一致性。

对于 KBP2017 的成绩，司罗表示：“很荣幸能够同全球的同行分享阿里巴巴的研究成果，人工智能在机器阅读理解和知识库构建上还处在起步阶段，我们正在积极和同行业顶尖机构学习交流，推动行业发展。比如我们内部建设的信息抽取平台 AliIE 项目就在同斯坦福大学展开积极合作”。

阿里巴巴正在将这样的信息抽取技术广泛的应用到实际业务当中，并致力于让更多的中小开发者从中收益。他们搭建的信息抽取平台 AliIE 拥有最顶尖的 AI 技术，并从一开始的架构设计就考虑到平台的开放性和可扩展性，可以让更多的开发者、研究员共同开发，并将成果回馈给这个社区。

阿里巴巴夺中文语法大赛 CGED 全球冠军

另一场比赛，中文语法错误自动诊断大赛（Chinese Grammatical Error Diagnosis，以下简称 CGED）由 IJCNLP 联办，今年已是第四届。比赛的背景是：学习中文的外国人数不断增加，由于中文的博大精深，外国友人在中文写作中会出现语法错误。 主办方挑选了一些外国友人写的中文作文片段，希望参赛者用人工智能算法自动识别里面的语法语义错误。

TEAM	RUNs	False Positive Rate	Detection Level				Identification Level				Position Level			
			Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
AL_I_NLP	run1	0.6172 (724/1173)	0.6439	0.686	0.7986	0.738	0.488	0.4791	0.5657	**0.5188**	0.2547	0.2169	0.2752	0.2426
	run2	0.6607 (775/1173)	**0.6465**	0.6792	**0.8284**	0.7464	0.4654	0.453	0.6006	0.5164	0.2264	0.1949	**0.2941**	0.2344
	run3	0.3052 (358/1173)	0.6173	0.7597	0.5714	0.6523	**0.5513**	0.6007	0.3756	0.4622	**0.4121**	**0.3663**	0.213	**0.2693**
BNU	run1	**0.098 (115/1173)**	0.4721	**0.7894**	0.2176	0.3411	0.4337	0.5474	0.106	0.1776	0.3775	0.2773	0.0418	0.0727
	run2	0.1355 (159/1173)	0.4794	0.758	0.2514	0.3776	0.4412	0.5527	0.131	0.2118	0.3735	0.2818	0.0515	0.0871
	run3	0.1893 (222/1173)	0.5181	0.7547	0.3448	0.4733	0.4696	0.5707	0.1786	0.2721	0.3798	0.2968	0.0715	0.1152
CVTE	run1	0.1441 (169/1173)	0.4756	0.7459	0.2504	0.3749	0.4461	**0.606**	0.1214	0.2023	0.3314	0.118	0.0204	0.0348
	run2	0.3154 (370/1173)	0.539	0.708	0.4528	0.5523	0.4711	0.5391	0.2057	0.2978	0.2602	0.1093	0.0465	0.0653
NTOUA	run1	1 (1173/1173)	0.6281	0.6281	1	**0.7716**	0.3211	0.3211	**0.6099**	0.4207	0.0212	0.0212	0.0958	0.0348
	run2	1 (1173/1173)	0.6281	0.6281	1	**0.7716**	0.3889	0.3889	0.506	0.4398	0.018	0.018	0.082	0.0295
YNU-HPCC	run1	0.6513 (764/1173)	0.5796	0.65	0.7163	0.6816	0.4218	0.4219	0.4217	0.4218	0.1778	0.1262	0.1191	0.1225
	run2	0.7383 (866/1173)	0.5891	0.6417	0.7829	0.7053	0.3879	0.3825	0.4575	0.4167	0.1426	0.1056	0.1191	0.112
	run3	0.6104 (716/1173)	0.5311	0.6298	0.6148	0.6222	0.3979	0.4086	0.3298	0.365	0.1702	0.0981	0.0698	0.0816

参赛机构比赛成绩公布

根据组委会公开的结果，司罗团队在所有的 3 个 level 的正确率都以较大优势位居第一，获取 2017 CGED 比赛的冠军。主力成员李林琳，谢朋峻，杨毅等通过在深度学习中引入无监督的语法知识，同时结合了集成学习等方法取得了好成绩。

司罗介绍，中文语法诊断的挑战性在于，中文语言知识丰富、语法多样；人在判断一句话是否有错误的时候，会用到长期积累的知识体系（比如一句话是否通顺、两个词是否可以搭配、语义上是否成立等）。相比之下，比赛提供的训练数据非常有限，仅通过训练数据来识别错误是很困难的。

赛题中包含的错误分为四种类型：多词（Redundant）、缺词（Missing）、错

词（Selection）和词序错误（Word Order）。系统性能的评估也由易到难分为 3 个 level：detection level（识别句子有没有错误）、identification level（识别错误句子的具体错误类型）和 position level（识别错误的位置和对应类型）

Table 1: Typical Error Examples.

Error Type	Original Sentence	Correct Sentence
M(missing word)	我河边散步的时候。	我在河边散步的时候。
R(redundant word)	流行歌曲告诉我们现在的我们的心理状态。	流行歌曲告诉我们现在的心理状态。
S(word selection)	还有其他的人也受被害。	还有其他的人也受伤害。
W(word order)	听多流行歌就会对唱那首歌的歌手痴迷。	多听流行歌就会对唱那首歌的歌手痴迷。

比赛要求诊断的四种错误类型

比如，“我要送給你一个庆祝礼物。要是两、三天晚了，请别生气”这句话，在第 3 个 Level，AI 需要明确指出“两、三天晚了”存在错误才能得分（正确用法应该是“晚了两、三天”）。

根据组委会公开的结果，司罗团队在所有的 3 个 level 的正确率都以较大优势位居第一，获取 2017 CGED 比赛的冠军。他们通过在深度学习中引入无监督的语法知识，同时结合了集成学习等方法。

技术细节上，IDST 团队在 bilstm-crf 模型的基础上，结合了分词、词性、依存句法等特征，同时将 language model 等无监督的知识 embedding 到神经网络。依靠 RNN 结构以及词性、依存等特征，不光能识别短程的语法错误，比如“一头牛”好于“一只牛”；也能识别比较长程的语法错误，比如“虽然父母很辛苦，而且对孩子照顾得很好”中“虽然”和“而且“不搭配。此外，他们针对比赛的 3 个不同 level，设计了不同的基于神经网络的 snapshot emsembles 方法。

司罗表示：“很荣幸能够同全球的同行分享阿里巴巴的研究成果，人工智能在对于自然语言的理解还处在起步阶段，要实现真正的语义理解还需要 5-10 年的跨越。我们正在积极和同行业顶尖机构学习交流，推动行业发展”。

司罗认为，自然语言处理是实现强人工智能的非常重要的一环，而且重要性会越来越显现。感知层面的事情越来越成熟了，认知层面也得跟上了。虽然有很大的鸿沟摆在面前，但这是必须要跨越的。“因为 NLP 技术是达到强人工智能的路上必须攻克的关键节点”。

如何让电脑成为看图说话的高手？计算机视觉顶会 ICCV 论文解读

阿里技术

阿里妹导读：ICCV，被誉为计算机视觉领域三大顶级会议之一。作为计算机视觉领域最高级别的会议之一，其论文集代表了计算机视觉领域最新的发展方向和水平。阿里巴巴在今年的大会上有多篇论文入选，本篇所解读的论文是阿里 iDST 与多家机构合作的入选论文之一，目标是教会机器读懂图片并尽量完整表达出来。

精准描述商品：计算机视觉和自然语言处理的联合

近年来，随着深度学习技术的快速发展，人们开始尝试将计算机视觉 (Vision) 和自然语言处理 (Language) 两个相对独立的领域联合起来进行研究，实现一些在过去看来非常困难的任务，例如“视觉 - 语义联合嵌入 (Visual-SemanticEmbedding)”。该任务需要将图像及语句表示成一个固定长度的向量，进而嵌入到同一个矢量空间中。这样，通过该空间中的近邻搜索可以实现图像和语句的匹配、检索等。

视觉语义联合嵌入的一个典型应用就是图像标题生成（Image Captioning）：对于任意输入的一张图像，在空间中找到最匹配的一句话，实现图像内容的描述。在电商场景下，淘宝卖家在发布一件商品时，该算法可以根据卖家上传得图片，自动生成一段描述性文字，供卖家编辑发布使用。再比如，视觉语义联合嵌入还可以应用于“跨模态检索（Cross-mediaRetrieval）”：当用户在电商搜索引擎中输入一段描述性文字（如“夏季宽松波希米亚大摆沙滩裙”、“文艺小清新娃娃领飞飞袖碎花 A 字裙”等），通过文字 - 图像联合分析，从商品图像数据库中找到最相关的商品图像返回给用户。

之前的不足：只能嵌入较短的语句简单描述图片

以往的视觉语义联合嵌入方法往往只能对比较短的句子进行嵌入，进而只能对图像做简单而粗略的描述，然而在实际应用中，人们更希望得到对图像（或图像显著区域）更为细致精确的描述。如图 1 所示，我们不仅想知道谁在干什么，还想知道人物的外表，周围的物体，背景，时间地点等。

图 1　现有方法的问题

现有方法："*A girl is playing a guitar.*"

我们提出的方法："*a young girl sitting on a benchis playing a guitar with a black and white dog nearby.*"

为了实现这个目标，我们提出一个框架：第一步从图像中找出一些显著性区域，并用具有描述性的短语描述每个区域；第二步将这些短语组合成一个非常长的具有描述性的句子，如图 2 所示。

图 2　我们的提出的框架

为此，我们在训练视觉语义联合嵌入模型时不仅需要将整个句子嵌入空间，更应该将句子中的各种描述性短语也嵌入空间。然而，以往的视觉语义联合嵌入方法通常采用循环神经网络模型（如 LSTM(Long short-term memory) 模型）来表示语句。标准的 LSTM 模型有一个链式结构（Chain structure）：每一个单元对应一个单词，这些单词按出现顺序排成一列，信息从第一个单词沿该链从前传到最后，最后一个节点包含了所有的信息，往往用于表示整个句子。显然，标准的 LSTM 模型只适合表示整个句子，无法表示一句话中包含的短语，如图所示。

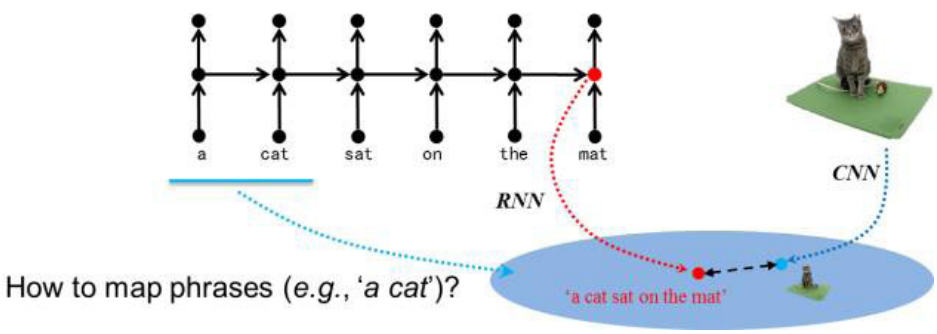


图 3　链式结构的问题

论文创新方法：提出层次化的 LSTM 模型

本文提出一种多模态、层次化的 LSTM 模型（Hierarchical Multimodal LSTM）。该方法可以将整个句子、句子中的短语、整幅图像、及图像中的显著区域同时嵌入语义空间中，并且自动学习出“句子 – 图像”及“短语 – 图像区域”间的对应关系。这样一来，我们生成了一个更为稠密的语义空间，该空间包含了大量的描述性的短语，进而可以对图像或图像区域进行更详细和生动的描述，如图所示。

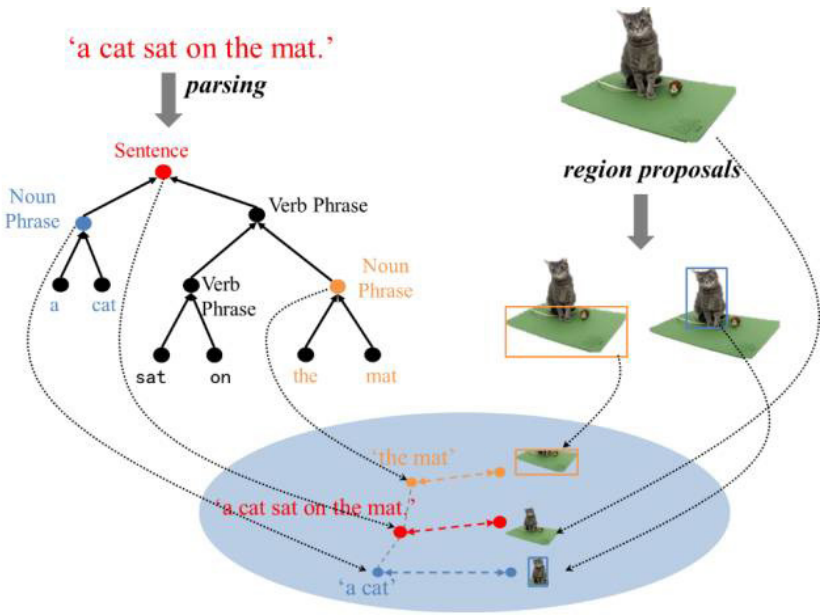


图 4　本文提出的多模态层次结构

本文方法的创新性在于提出了一个层次化的 LSTM 模型，根节点对应整句话或整幅图像，叶子节点对应单词，中间节点对应短语或图象中的区域。该模型可以对图像、语句、图像区域、短语进行联合嵌入（Joint embedding），并且通过树型结构可以充分挖掘和利用短语间的关系（父子短语关系）。其具体网络结构如下图所示

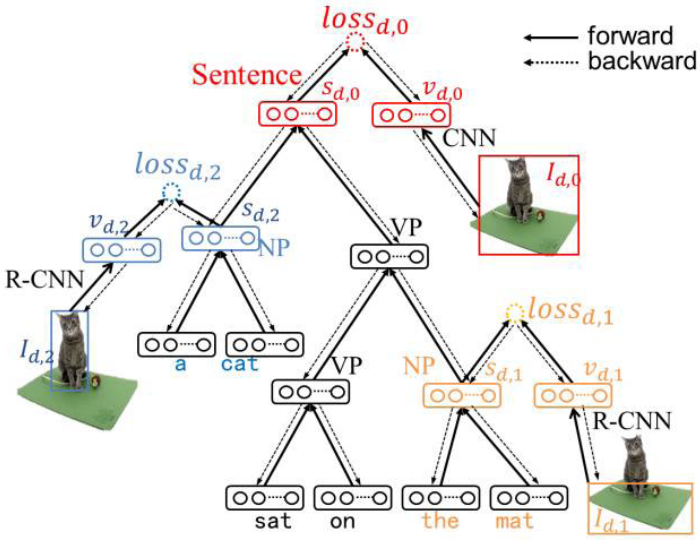


图 5　网络结构

其中为每一个短语和对应的图像区域都引入一个损失函数，用于最小化二者的距离，通过基于结构的反向传播算法进行网络参数学习。

在图像 – 语句数据集上的比较

	Image Annotation			Image Search		
	R@1	R@10	Med *r*	R@1	R@10	Med *r*
SDT-RNN	9.6	41.1	16	8.9	41.1	16
DeFrag	14.2	51.3	10	10.2	44.2	14
SC-NLM	14.8	50.9	10	11.8	46.3	13
DeepVS	22.2	61.4	4.8	15.2	50.5	9.2
NIC	17.0	56.0	7	17.0	57.0	7
m-RNN-vgg	35.4	73.7	3	22.8	63.1	5
DeepSP	35.7	74.4	N/A	25.1	66.5	N/A
Ours	38.1	76.5	3	27.2	68.8	4

图 6　在 Flickr30K 数据集上的对比

	Image Annotation			Image Search		
	R@1	R@10	Med *r*	R@1	R@10	Med *r*
Random	0.1	1.1	631	0.1	1.0	500
DeepVS	36.4	80.9	3	28.1	76.1	3
m-RNN	41.0	83.5	2	29.0	77.0	3
DeepSP	40.7	85.3	N/A	33.5	83.2	N/A
Ours	43.9	87.8	2	36.1	86.7	3

图 7　在 MS-COCO 数据集上的对比

可见本文方法在几个公开数据集上都获得了很好的效果

在图像区域 – 短语数据集上的对比

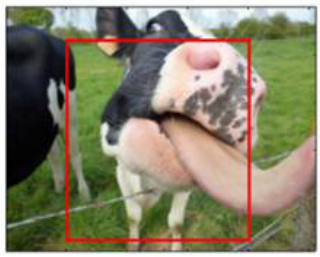
我们提供了一个带有标注的图像区域 – 短语数据集 MS-COCO-region，其中人工标定了一些显著性物体，并在这些物体和短语之间建立了联系。

	Region Annotation			
	R@1	R@5	R@10	Med *r*
Random	0.02	0.12	0.24	3133
DeepVS	7.2	18.1	26.8	64
m-RNN	8.1	20.6	28.2	56
Ours	10.8	22.6	30.7	42

图 8　在 MS-COCO-region 数据集上的对比

下图是我们方法的可视化结果，可见我们的短语具有很强的描述性

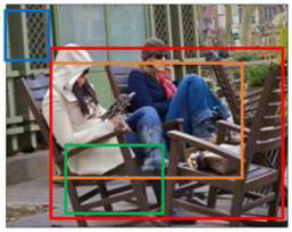
(1) *a white and gray cat with a striped tail*
(2) *a white and gray cat*
(3) *a cat with an intent look*



(1) *a cow standing in the grass with a tag in its ear*
(2) *a cow with a black face*
(3) *a cow staring into the camera*
(4) *a close up of a black and white cow*

此外，我们可以学习出图像区域和短语的对应关系，如下

"*Two people sitting on rocking chairs on the deck.*"



(1) *two people*
(2) *rocking chairs*
(3) *the deck*
(4) *two people sitting on rocking chairs*

"*An empty room containing a plant and a painting on the wall.*"

(1) *an empty room*
(2) *a plant*
(3) *wall*
(4) *a painting*
(5) *a plant and a painting*

机器学习

揭秘支付宝中的深度学习引擎：xNN

弘川

阿里妹导读：本文介绍支付宝 App 中的深度学习引擎——xNN。xNN 通过模型和计算框架两个方面的优化，解决了深度学习在移动端落地的一系列问题。xNN 的模型压缩工具 (xqueeze) 在业务模型上实现了近 50 倍的压缩比，使得在包预算极为有限的移动 App 中大规模部署深度学习算法成为可能。xNN 的计算性能经过算法和指令两个层面的深度优化，极大地降低了移动端 DL 的机型门槛。

深度学习——云端还是移动端?

近来，深度学习（DL）在图像识别、语音识别、自然语言处理等诸多领域都取得了突破性进展。DL 通常给人以计算复杂、模型庞大的印象——从 Siri 语音助手到各种聊天机器人、再到支付宝“扫五福”，移动端收集数据 + 云端加工处理似乎成为一种常识。然而对很多应用来说，这种模式其实只是无奈之选。

去年春节的“扫五福”活动中，为了识别手写“福”字，支付宝多媒体团队调动了近千台服务器用于部署图像识别模型。可是如此规模的集群也没能抵挡住全国人民集五福的万丈热情。为了防止云端计算能力超载，活动中后期不得不启动了降级预案——用计算量小但精度也较低的传统视觉算法替代了 DL 模型。降级虽然不妨碍大伙继续热火朝天地收集福卡，但对用户体验无疑是有一定影响的，比如一些不可言说的汉字也被误判成了“福”字。

另一方面，DL 在云端则意味着数据必须上传。即使不考虑计算压力，从网络延时、流量、隐私保护等角度也给用户体验带来种种限制。因此，对相当多的应用来说，DL 模型前移到移动端部署可以看作是一种刚需。

两大挑战

最近，随着手机处理器性能的提升和模型轻量化技术的发展，移动端 DL 正在变得越来越可行，并得到了广泛的关注。苹果和谷歌已经分别宣布了各自操作系统上的 DL 框架 Core ML 和 Tensorflow Lite，这无疑将极大地促进移动端 DL 的发展。但是，尤其对于支付宝这样的国民 App 来说，仍然存在一些严峻的挑战是无法通过直接套用厂商方案来解决的。

1. 机型跨度大：支付宝 App 拥有数亿受众群体，在其中落地的业务必须对尽可能多的用户、尽可能多的机型提供优质的体验。对支付宝来说，参考 Core ML 只将功能开放给少数高端机型的做法是不合适的。因而无论在运行速度和内存占用等性能指标、还是在兼容性上，支付宝的移动端 DL 都必须做到极致，才能最大幅度地降低使用门槛。

2. 包尺寸要求严：支付宝 App 集成了众多的业务功能，安装包资源非常紧张，一个新模型要集成进安装包往往意味着需要下线其他的功能。而即便通过动态下发的形式进行部署，DL 模型的大小也会强烈影响用户的体验。随着移动端智能化程度的不断提升，直接在端上运行的 DL 应用必然会越来越多，这以当前单个模型大小就动辄数十、数百 M 的尺寸来看几乎是不可想象的。同时，移动端 DL 引擎本身的 SDK 也需要尽可能地瘦身。

五大目标

支付宝 xNN 是针对国民 App 环境定制开发的移动端 DL 解决方案，项目制定了如下技术目标。

1. 轻模型：通过高效的模型压缩算法，在保证算法精度的前提下大幅减小模型尺寸。

2. 小引擎：移动端 SDK 的深度裁减。

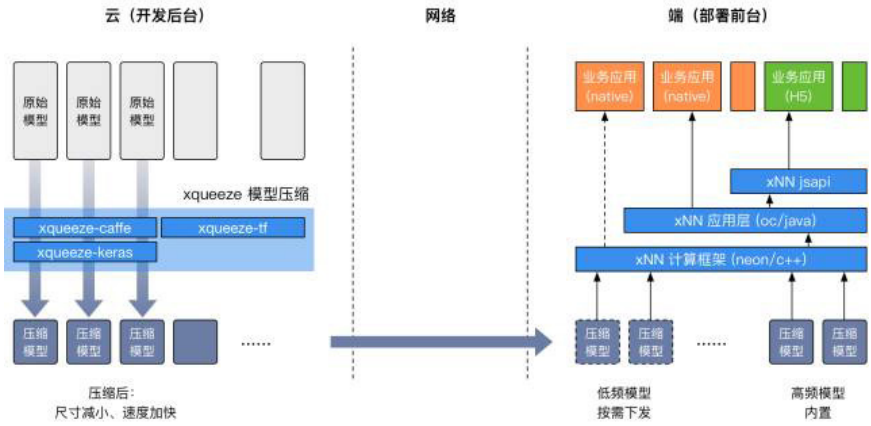
3. 快速：结合指令层和算法层的优化，综合提升 DL 计算的效率。

4. 通用：为保证最大的机型覆盖率，以最为通用的 CPU 而非性能更强劲的 GPU 作为重点优化平台。不仅支持经典的 CNN、DNN 网络，也支持 RNN、LSTM 等网络形态。

5. 易用：工具链对业务保持高度友好——使得算法工程师们能更好地专注于算法本身，在不需要成为模型压缩专家和移动端开发专家的情况下都能快速完成云端模型到移动端模型的转换和部署。

主要特性一览

xNN 为 DL 模型提供了从压缩到部署、再到运行时的统计监控这一全生命周期的解决方案。xNN 环境由开发后台和部署前台两部分组成。



开发后台以 xqueeze 工具链为核心，支持多种训练框架。业务可以使用 xqueeze 压缩、优化自己的 DL 模型，得到尺寸大幅减小、运行速度显著加快的模型版本。压缩后的模型根据使用场景，可以通过 App 安装包内置或按需下发的形式部署到移动端。

在部署前台，xNN 的计算框架提供高效的前向预测能力。xNN 的应用层在计算的基础上还提供了模型下发、数据统计、错误上报等一站式能力。xNN 还通过一个 jsapi 提供了直接对接 H5 应用的能力——通过 DL 模型的动态下发和 H5，能够实现

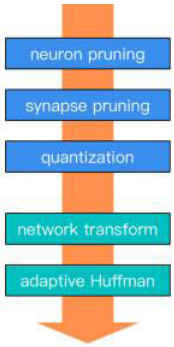
完全的动态化，从而在客户端不发版的情况下完成算法 + 逻辑的同时更新。

	xNN (支付宝)	core ML
模型压缩	xqueeze	不支持
稀疏运算	支持	不支持
CNN	支持	支持
目标检测	SSD	不支持
LSTM	支持	支持
Depthwise CNN	支持	支持
训练框架	Caffe, Tensorflow, Keras	Caffe, Keras
部署框架	兼容Caffe/私有	私有
模型更新	支持	不支持
支持机型	无限制	iPhone ≥6S/SE

上图给出了 xNN 的主要特性。在 xqueeze 模型压缩的基础上，xNN 还支持通过快速处理稀疏网络来提高性能。xNN 支持了丰富的网络结构类型，包括经典 CNN/DNN、SSD 目标检测和 LSTM。xNN 的部署框架原生兼容 Caffe，业务可以在不做转换的情况下直接在移动端运行已有的 Caffe 模型，以快速评估效果。而经过压缩的私有格式模型更小、更快。在 Tensorflow 和 Keras 平台上训练的模型也能够在原有的环境上进行压缩，然后转换为 xNN 支持的格式部署到移动端。不同于 core ML，xNN 理论上支持安卓和 iOS 上的所有机型。

xqueeze 模型压缩

xNN-xqueeze 的模型压缩流程如下图之 (a) 所示，包括神经元剪枝 (neuron pruning)、突触剪枝 (synapse pruning)、量化 (quantization)、网络结构变换 (network transform)、自适应 Huffman 编码 (adaptive Huffman)、共 5 个步骤。其中前三步理论上是有损的，而使用 xqueeze 对网络权重和压缩超参进行 finetune，能够将精度的下降保持在可控甚至可忽略的程度。后两步则完全不影响网络的输出精度。整个流程不仅会减小模型的尺寸，还通过网络的稀疏化和结构优化，显著提高前向预测的速度。



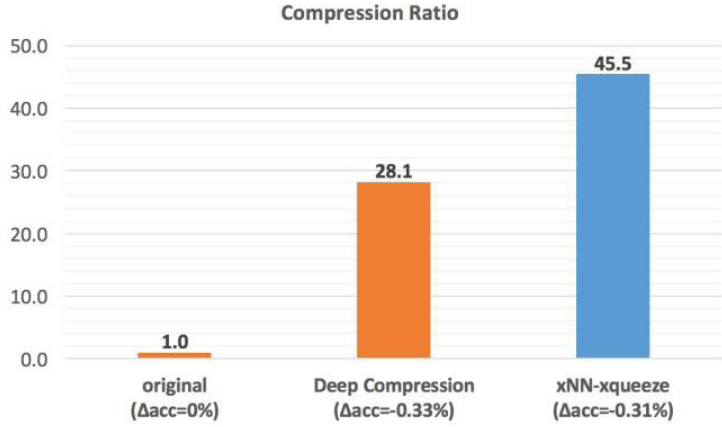
(a) xNN-xqueeze flow

	xNN-xqueeze (支付宝)	Deep Compression
Neuron Pruning	支持	不支持
Synapse Pruning	支持	支持
Quantization	支持	支持
Network Transform	支持	不支持
Entropy Coding	自适应Huffman	游程+Huffman

(b) xNN-xqueeze vs "Deep Compression"

在领域的经典方案 DeepCompression 的基础上，xqueeze 进一步扩充了 neuronpruning 和 network transform 的能力。其中，neuron pruning 能够逐次裁剪掉“不重要”的神经元和与之对应的权重参数。通过 neuron pruning 和 synapse pruning 的结合，在模型精度和压缩比之间达成更好的平衡。xqueeze 还具有 network transform——在网络的宏观层面进行优化的能力，networktransform 脚本扫描整个网络，诊断出可优化的点，包括在有条件的情况下自动地进行层 (layer) 的组合与等效替换。此外，xqueeze 通过自适应地使用 Huffman 编码，有效提升不同稀疏程度的模型之压缩比。

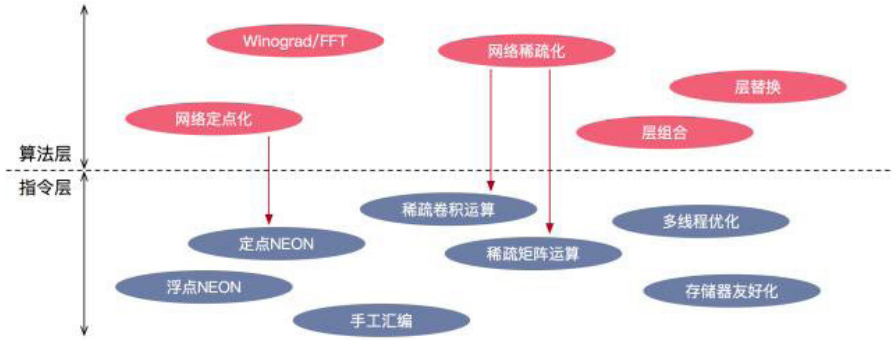
如下图所示，对于业务分类模型，使用 xqueeze 工具链能够实现 45.5 倍的压缩，在同等程度的精度损失下，压缩率超越经典方案达 60%。



xNN 计算性能优化

xNN 的性能优化不局限于底层，而是通过与 xqueeze 工具链的配合，在算法和指令两个层面同步发力，为更为深入的优化创造空间。

如下图所示，在算法层，xqueeze 的剪枝在压缩模型尺寸的同时，也促进了网络的稀疏化——即催生出大量的零值权重。相应地，xNN 在指令层实现了稀疏运算模块，在卷积和全连接计算中，自动忽略这些零值权重，减小计算开销，提升速度。又如之前已经提到的，在 xqueeze 的 network transform 阶段，会对网络进行宏观层面的优化，包括将相邻的层进行结果上等效的组合与替换，来减少计算的冗余度和提高访问存储器的效率。要充分发挥 network transform 的效能，也离不开指令层实现的支持。



在指令层，xNN 通过智能调配各个内核的负载，提升多线程环境下的性能。xNN 在设计中不仅关注计算资源，还充分考虑了访问存储器的开销，通过精细化地调度数据的读写来提升 cache 的命中率。最为重要的是，所有核心计算模块均由某位芯片行业出身的指令集架构专家一条一条汇编代码手写而成。

在以 SqueezeNet 为基础的业务分类模型上，xNN 在 Qualcomm 820 CPU 上能够输出 29.4 FPS 的前向预测帧率，在苹果 A10 CPU (iPhone 7) 上的帧率则达到 52.6 FPS，比 CPU 与 GPU 并用的 Core ML 还要更快。

业务落地

支付宝 App 已经集成了 xNN。在支付宝的“AR 扫一扫”入口，90% 以上的

Android 和 iOS 机型都在使用 xNN 来完成前置物品分类，向用户推荐“AR 扫花识花”等便利功能。xNN 本身的健壮性也经受住了“七夕送你一朵花”这样高强度、广机型覆盖的大型运营活动的考验。该模型的最新版本在确保精度的前提下，尺寸已压缩到 100KB 以下。Android 平台上，全功能 xNN 的 SDK 包增量仅 200KB 出头，若根据特定应用做裁剪，将能够轻松减小到 100 多 KB。

xNN 上线后，已在蚂蚁和阿里内部引起了强烈反响，一大波移动端 DL 应用正在基于 xNN 紧张开发中，并在未来的几个月中逐步提供给用户使用。

求贤若渴

最后进入本文重点！

如果你对最前沿的移动端 DL 技术充满热情，希望一起来开发牛逼的 DL 引擎和牛逼的 DL 模型，我们已经为你准备好了可能是业界最优秀的应用场景——数亿用户的国民 App、最广阔的发展空间——蚂蚁金服集团、和最为靠谱的队友——由来自软硬件各领域的资深工程专家、博士、博士后、教授们组成的技术团队！

如何用机器学习方法，提升另一半的满意指数?

阿里技术

阿里妹导读：今天是七夕情人节，我们来探讨一个严肃又甜蜜的重要问题，一个你可能正在关注、或者终要关注的人生课题：如何用机器学习方法，为你生命里的另一半，挑选最适宜的母婴产品，提升幸福满意指数。

背景介绍

生命阶段在消费行为中的重要作用已经在营销和社会学中被研究了几十年。虽然这些研究并没有关注消费者的行为，但是他们研究了各种人和事件的生命周期，这为研究生命阶段对消费者行为的影响提供了坚实的基础。在电子商务中，比起用户的生命阶段转变，更多的研究侧重于根据消费者的历史行为进行商品推荐。

例如，电商公司会挖掘具有相似偏好的用户，并根据其他相似用户的偏好对当前用户进行推荐。而对于消费者生命阶段与消费者行为之间关系的研究才刚刚开始，在这些研究的驱动下，我们提出了一种可用于电子商务中生命阶段推断的动态融合算法。

我们使用多元逻辑回归模型对婴儿的生命阶段进行分类预测、产生相应的概率分布。此外我们还开发了动态融合方法以不断提高预测精度，并且可以有效地提高计算

效率。每次有新的概率分布生成后，我们会更新然后维护多个概率分布。这样做可以识别消费者的短期兴趣，而且对于多个孩子的生命阶段预测也是非常有帮助的。

为了评估算法的有效性，我们进行了大量离线和在线的数值实验，这些实验表明我们的方法可以显着提高消费者生命阶段推断的准确性。

本文的主要贡献有：

1. 我们为生命阶段推断提供了工业级别的解决方案。

2. 我们开发了一种动态融合方法，可以在大幅节省计算资源的基础上不断提高预测精度，并且可以方便地维护对多个孩子年龄阶段的预测。

3. 我们通过实际数据验证了我们的解决方案对生命阶段推断的有效性。

母婴用户生命阶段划分

用户行为会随着生命阶段而改变，消费行为的转变通常与生命阶段的转变一致，这种现象在母婴这样的垂直行业中更为显著。例如，妈妈们会在婴儿刚出生时购买尿布；而在 2 到 3 年后，当婴儿要上幼儿园的时候，妈妈们会购买更多的衣服和鞋子。消费者行为随着生命阶段改变的现象不仅仅存在与母婴行业，在其他行业例如家装和汽车也有相同的现象。在本文中，我们将重点关注母婴行业，即基于父母的消费行为推断婴儿的生命阶段。

根据我们的行业认知，一个孩子的生命阶段发展是一个与年龄密切相关的连续过程。因此我们将母婴用户的生命阶段根据孩子的年龄分为以下几个阶段：出生前（妈妈的孕期）；0–6 个月（新生儿）；6–1 个月；2–3 岁（托儿所）；3–7 岁（幼儿园）。不同年龄段孩子的父母会对不同的商品感兴趣，如果我们可以准确预测他们孩子的年龄段并推荐合适的商品，就可以大幅提高转化率。

动态融合方法

为了推断一个孩子的生命阶段，我们开发了一种不断预测并且不断改进推论的算法。与使用复杂模型进行一次性预测的方法不同，我们的算法每次会根据当前的数据产生一个较好的预测结果，然后不断地更新我们的推论，这就是所谓的动态融合过程。下面将会具体介绍动态融合过程的细节。

与单个预测结果相比，对孩子生命阶段预测的概率分布包含了更多的信息。比如说当分布中有两个生命阶段的概率都较高时，这表明消费者可能有两个孩子，或者消费者的孩子正处于两个生命阶段的交界处，但我们不能从单个预测结果中得到这些信息。保留这些概率分布可以推断用户孩子的生命阶段，并且可以让我们在合适的时候更新推论。然而消费者在不同月份的行为很可能导致不同分布中概率最高的生命阶段是不同的，如何维护和更新这些分布成了我们解决方案的关键。因此我们设计了动态

融合算法来解决这个问题。

以特征向量 X 作为输入，我们可以通过模型预测单月概率分布，我们将在下一节中介绍模型的训练细节。现在假设我们已经有$P(y = k|X)$的概率分布，并且当下个月结束时，模型会产生另一个概率分布$P'(y = k|X')$。有了这两个分布后，为了将它们进行融合，首先需要将之前的分布进行平移，平移方式由下式给出：

$$P(y = k|X) = P_d(y = k + \Delta|X) \quad (1)$$

其中 Δ 是之前分布产出时间与当前月份的时间差。平移概率分布的方法有好几种，在 Algorithm 1 中对我们使用的方法进行了详细介绍。

然后我们会比较平移后的分布，如果两个分布具有最高概率的生命阶段是相同的，也就是说：

$$argmax_k\ P'(y = k|X') = argmax_k\ P_d(y = k|X)$$

这样的话我们就可以将这两个分布融合在一起：

$$P_{new}(y = k|X') = \frac{P_d(y=k|X) \times P\prime(y=k|X\prime)}{\sum_{i=1}^{K} P_d(y=i|X) \times P\prime(y=i|X\prime)} \quad (2)$$

其中$\sum_{i=1}^{K} P_d(y = i|X) \times P'(y = i|X')$用于对新分布进行归一化。如果多个分布概率最高的生命阶段不同，那多个分布都会被保留以便将来可能的融合。我们会记录每个分布融合的次数，推断的生命阶段由融合次数最多的分布决定。当新的单月分布生成时，将按照相同的逻辑进行算法的下一次迭代。

特征工程

在电商场景中，所有的特征都来自消费者的五大类行为：搜索、点击、收藏、加购和购买。我们使用的特征分为以下几类：

1. 类目特征

在淘宝的类目体系中存在多级类目结构，其中一级类目包括衣服、鞋子等主要的大类；类目级别最多可达到 4 个或 5 个，没有子类目的最低级别类目称为叶子类目。理论上我们可以使用商品 ID 作为特征，但是这会导致特征矩阵过于稀疏，只有极少

量的样本会包含某些特征。为了避免这种情况，同时仍然能捕捉到用户不同的消费兴趣，我们使用商品对应的一级类目和叶子类目作为特征。

2. 类目属性特征

相同类目下的商品会共享一些属性：例如商品的品牌属性可能是 IBM、New Balance 等具体品牌，商品的尺寸属性可以是“S”、“M”或“L”。类目属性特征是指商品类目和属性的组合特征，我们把母婴行业中所有的类目属性特征作为模型的输入。

Algorithm 1 A dynamic merging process

Require: X, sets $\{S_k\}$ as Null, $k = 1 \dots K$

if $\{S_k\}$ is not null **then**
  for each distribution P in S, **do**
    Disaggregate P into months: $P_t(y = k|X) = \frac{P(y=k|X)}{|T|}$ where $|T|$ is the number of months in life stage k
    Move P_t by months: $P^d_{t+\delta}(y = k|X) = P_t(y = k|X)$ and for $P^d_t(y = k|X)$ where $t < \delta$, fill in its value by 0.0001 to avoid numerical issues.
    Aggregate P_t in to P of life stages: $P(y = k|X) = \sum_{t=1}^{|T|} P_t(y = k|X)$, where $|T|$ is the number of months of life stage k. And for $P(y = K|X)$, we also need to add additional probabilities between $P_{|T|}(y = K|X)$ and $P_{|T|+\delta}(y = K|X)$.
    Add P into S_k based on the life stage with the highest probability.
  end for
end if
if X is not empty **then**
  Generate $P_{new}(y = k|X)$, where $k = 1, \dots, K$
  Add P_{new} into S_k where k $= \underset{i}{argmax} P_{new}(y = i \mid X')$
end if
for each S_k, **do**
  merge all the distributions in S_k to generate a new $P(y = k|X)$ by $P(y = k|X) = \frac{P(y=k|X) \times P(y=k|X)}{\sum_{i=1}^{K} P(y=i|X) \times P'(y=i|X)}$, and add up merge times of each original distributions as the new merge times for $P(y = k|X)$.
end for

16: Find the distribution P with the largest merge times and record the life stage k_1 with the largest probability.
17: **if** the distribution P' with the second largest merge times takes up more than 60% of the largest merge times, **then**
18: Conclude that this family may have a second child and the life stage k_2 is one from P' with the highest probability
19: **end if**
20: **for** each S_k, **do**
21: restore $P_t(y = k|X)$ to the distribution before diplaced if it is not merged with any other distribution
22: **end for**
return k_1 and k_2, where k_1 and k_2 exist.

3. 商品属性特征

除了类目属性特征，我们还将商品本身的属性作为输入特征。

4. 搜索词特征

搜索词是指用户用于搜索的关键词，可能会直接对应孩子的年龄段，例如“3 岁宝宝的衣服”、“3 段奶粉”等，我们会选取一些特定的关键词作为输入特征。

5. 商品标题特征

商品标题包含了丰富的信息，其中可能也包括年龄段或者生命阶段相关的信息。我们整理了大约 200 个年龄段相关的关键词，对标题进行相关处理后作为输入特征加入模型。

6. 时序特征

消费者在不同日期购买相同的商品也有不同的含义，一个妈妈在 6 个月前购买尿布和在 1 周前购买尿布的含义是不同的。因此不同月份的行为也被归为不同的特征，为了减少模型训练时的计算负担，我们使用了用户最近一个月的所有行为和去年同月的购买行为作为输入特征。

数值实验

为了证明我们设计的方法是有效的，我们利用半年的 taobao.com 的数据做了多组实验 (从 2016 年 9 月到 2017 年 2 月)。图一和图二分别简要说明了训练数集和测试数集中婴儿生命阶段的分布。

Figure 1: The distribution of children's age in the training set

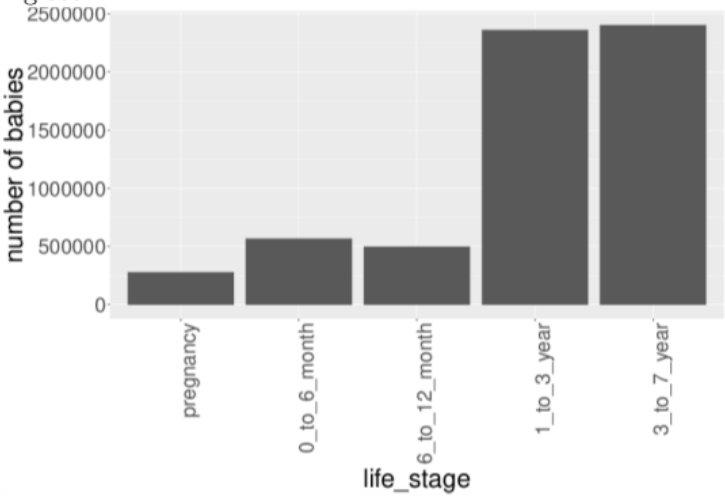
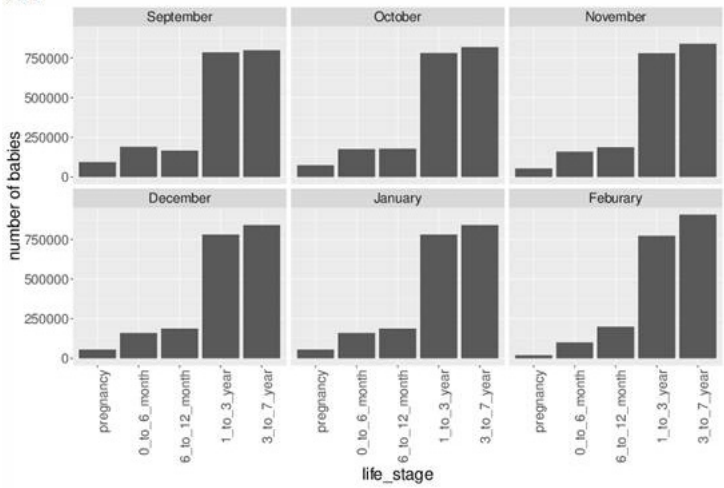


Figure 2: The distribution of children's age in the testing set



值得注意的是，随着时间的推移，婴儿会逐渐长大，所以生命阶段的分布会沿着时间轴向右移动。实验中，我们先使用 2016 年 9 月份的数据进行预测。得到的生命阶段会和下一个月的生命阶段的预测根据公式（2）融合在一起。融合之后的结论会作为 10 月份关于生命阶段的最后结论。而这个结论会继续和下一个月单月数据预测的结论（利用 2016 年 11 月的特征数据得到的生命阶段的结论）融合。相应的，利用单月特征得到的预测结论作为对照组实验的结果。我们将对照组实验使用的方法称为无记忆方法。也即是说这种方法仅仅使用最近一个月的特征数据来进行推断。

表 2，3，4 展示了从 2016 年 9 月份到 2017 年 2 月份的实验结果。值得注意的是因为 2016 年 9 月份作为起始点，所以两种方法的结果是一样的。正如表 2 和表 3 所展示的，基于动态融合的方法超越了。

表 2，3，4 展示了从 2016 年 9 月份到 2017 年 2 月份的实验结果。值得注意的是因为 2016 年 9 月份作为起始点，所以两种方法的结果是一样的。正如表 2 和表 3 所展示的，基于动态融合的方法几乎比无记忆方法好 10%。表 4 展示了不同月份的比较结果。5 个月（2016 年 9 月份不包含在比较结果中）中有 2 个月动态融合方法在准确度显著超越了无记忆方法（最高达 15%），在另外的 3 个月中无记忆方法比动态融合方法预测的更准确（不超过 10%）。动态融合方法在召回率上超越了无记忆方法。5 个月中，动态融合方法在 3 个月中超越了无记忆方法，最高记录高达 21.76%，而在另外 2 个月中只比无记忆方法稍微逊色，最坏记录不超过 −1%。

这背后的原因在于消费者的行为强度在一年中的旺季和淡季变化剧烈。对于任何一个拥有丰富的消费者行为数据的月份，两种方法的准确度十分相似。然后，对于任何一个消费者行为数据十分稀疏的月份，无记忆方法的准确度会明显下降，然后动态融合方法可以通过融合消费者行为丰富的月份中得到的结论这样的方式巧妙的克服上述缺点。

Life stages	user_ count	m atch	inference	recall	accuracy
Pregnancy	84,434	53,642	63,999	63.53%	83.82%
0 to 6 months	186,732	142,519	174,457	76.32%	81.69%
6 to 12 months	164,780	122,706	154,458	74.47%	79.44%
1 to 3 years	781,756	595,802	687,604	76.21%	86.65%
3 to 7 years	770,488	701,933	907,672	91.10%	77.33%
0 verall	1,988,190	1,616,602	1,988,190	81.31%	81.31%

Table 1: Statistics of inference for testing sets of Septem - ber, 2016

Life stages	user_ count	m atch	inference	recall	accuracy
Pregnancy	13,157	3,990	6,721	30.33%	59.37%
0 to 6 months	96,645	73,090	113,734	75.63%	64.26%
6 to 12 months	197,431	116,184	136,047	58.85%	85.40%
1 to 3 years	768,917	485,080	556,462	63.09%	87.17%
3 to 7 years	884,417	839,320	1,147,603	94.90%	73.14%
0 verall	1,960,567	1,517,664	1,960,567	77.41%	77.41%

Table 2: Statistics of inference for testing sets of Febru- ary 2017 by D ynam ic M erging A pproach

Life stages	user_ count	m atch	inference	recall	accuracy
Pregnancy	13,157	4874	11815	39.03%	41.25%
0 to 6 m onth	96,645	65611	15414	73.12%	42.46%
6 to 12 m onth	197,431	61809	90359	34.21%	68.40%
1 to 3 year	768,917	358210	435947	51.41%	82.17%
3 to 7 year	884,417	714690	1055640	92.99%	67.70%
O verall	1,960,567	1205194	1748275	68.94%	68.94%

Table 3: Statistics of inference for testing sets of February 2017 by M em oryless A pproach

Table 4: T he accuracy and recall rate of the dynam ic m erging approach

M onths	A ccuracy D M A	A ccuracy M A	R ecall D M A	R ecall M A	D ifference in accuracy	D ifference in recall
Sep tem b er	81.31%	81.31%	81.31	81.31%	0%	0%
O ctob er	69.40%	78.85%	69.40%	70.02%	-9.45%	-0.66%
N ovem b er	66.82%	70.26%	66.82%	62.77%	-3.44%	4.05%
D ecem b er	70.05%	76.29%	70.05%	70.80%	-6.24%	-0.75%
January	76.51%	61.98%	76.51%	54.75%	14.53%	21.76%
February	77.41%	68.94%	77.41%	61.47%	8.53%	15.94%

D M A : D ynam ic M erging A pproach
M A : M em ory less A pproach

实验的月份越多，关于生命阶段的结论会因为越来越多的消费者行为被隐性地考虑在模型中而变得越稳定。这个方法的另外一个优势在于，对于任何一个错误的推断可以在接下来的月份中得到修正。这一点可以被实验中 2016 年 11 月、12 月，2017 年的 1 月和 2 月份的结果说明。11 月份中错误的结论在接下来的月份中被逐渐修正。读者会质疑正确的结论也可能被改为错误的结论，但是我们的实验表明总体

上，动态融合方法不会差于无记忆方法。

结论

在本篇论文中，我们介绍了一种叫做动态融合的创新的方法用来基于家长的消费行为推断幼儿的生命阶段。我们详细介绍了特征工程，以及算法的细节。我们还进行了计算实验用于证明这个算法的优势。

对于消费者行为数据丰富的月份，该方法没有比对照组差，然而在消费者行为数据稀疏的月份，该方法可以超越对照组方法。在未来的工作中，我们可以探索其他的机器学习模型用于预测单月的行为，或者多层级的模型来继续提高表现（例如一种机器学习模型用于预测某些特殊的生命阶段而另外一种机器学习模型用于预测其他生命阶段）。此外，该生命阶段的预测还可以被其他应用场景（如推荐系统）使用。

团队：新零售供应链平台事业部—技术部—数据决策

如何搭建大规模机器学习平台？以阿里和蚂蚁的多个实际场景为例

阿里技术

阿里妹导读：近年来，随着“大”数据及“大”模型的出现，学术界和工业界对分布式机器学习算法引起了广泛关注。针对这一刚需，本论文设计了一个独一无二的分布式平台——**鲲鹏**。它无缝的结合了分布式系统及并行优化算法，解决了大规模机器学习算法带来的一系列问题。鲲鹏不仅囊括了数据 / 模型并行、负载平衡、模型同步、稀疏表示、工业容错等特性，而且还提供了封闭好的、宜于调用的 API 供普通的机器学习者开发分布式算法，降低使用成本并提升效率。

本论文的实验在十亿级别的样本和特征数据上进行，结果表示，鲲鹏这一设计使得一系列算法的性能都得到了极大的提升，包括 FTRL，Sparse-LR，以及 MART。此外，鲲鹏在阿里巴巴双 11 狂欢购物节及蚂蚁金服的交易风险检测中体现出了其巨大的应用价值。

研究背景

现在是个大数据的时代，各个平台的数据量都与时俱进。举例而言，国外的 Twitter 每天新增 5 亿条 Tweets，阿里巴巴每天有 5000 万个包裹，蚂蚁金服的支付宝交易峰会达到 12 万笔 / 秒，仅仅在 2016 年双 11 当天就产生了 10.5 亿条交易。如此大的数据量使得机器学习不得不面临着样本及特征规模巨大的挑战。例如，阿里巴巴内部的模型会达到千亿样本，百亿特征，TB-TP 级的训练数据量。因此，如果搭建能够训练如此大规模数据的机器学习平台是工业界面临的一个巨大问题。

已有方法介绍

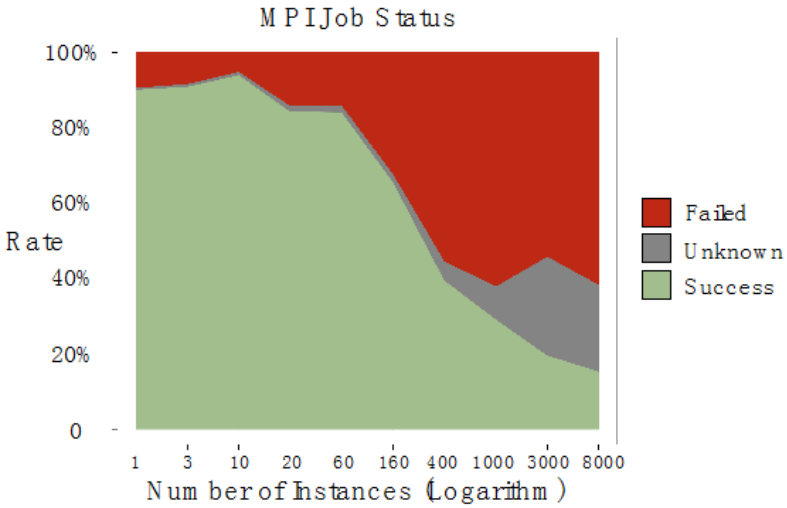


图 1　阿里某生产集群中 MPI 任务状态

目前，业界已经有一些比较成熟的分布式处理框架，如 Hadoop，Spark，GraphLab 和 GraphX。虽然它们可以支持机器学习算法并行化，但它们很难让开发人员设计出更有效率且支持更大规模的机器学习算法。具体而言，Hadoop 和 Spark 虽然提供了一些同步和粗粒度运算符（例如，Map，Reduce 和 Join 等），但主要还停留在解决中小规模机器学习的问题。

GraphLab/GraphX 主要是为了图存储和计算，并不适用于普通的大规模机器学习算法。MPI 虽然能够支持普通的分布式计算，但其缺乏容错机制。特别是在

worker 很大的情况下，MPI 的运行成功率会大大降低，如图 1 所示。因此，如何设计更有效率且支持更大规模的机器学习算法成为一个业界难题。

鲲鹏的研究动机及创新性

鲲鹏取名自《庄子 · 逍遥游》, 文中记载“北冥有鱼，其名曰鲲。**鲲**之大，不知其几千里也；化而为鸟，其名为**鹏**。鹏之背，不知其几千里也。怒而飞，其翼若垂天之云。”在我们的鲲鹏系统中，“鲲”即是超大规模分布式计算系统，它拥有超强的计算能力；而“鹏”即是超大规模分布式优化算法，它建立在“鲲”之上。“鲲鹏”即同时拥有超大规模分布式计算系统及超大规模分布式优化算法，合二为一使得它有“一飞冲天”的能力，如图 2 所示。

图 2　鲲鹏的研究动机及创新性

系统创新

鲲鹏的创新在于它拥有了以下功能：

1. 强大的容错功能，甚至在复杂且忙碌的线上集群环境中
2. Backup Instancefor Straggler Management
3. 支持有向无循环图形式的调度和同步，包括 BSP/SSP/ASP
4. 用户友好的界面和编程

算法创新

鲲鹏架构使得常用的机器学习算法的大规模化成为了可能，截止目前，已经有众多机器学习算法在鲲鹏上得以实现和应用，包括但不限于 LR，FTRL，MART，FM，HashMF，DSSM，DNN，LDA。

鲲鹏的架构

总体架构

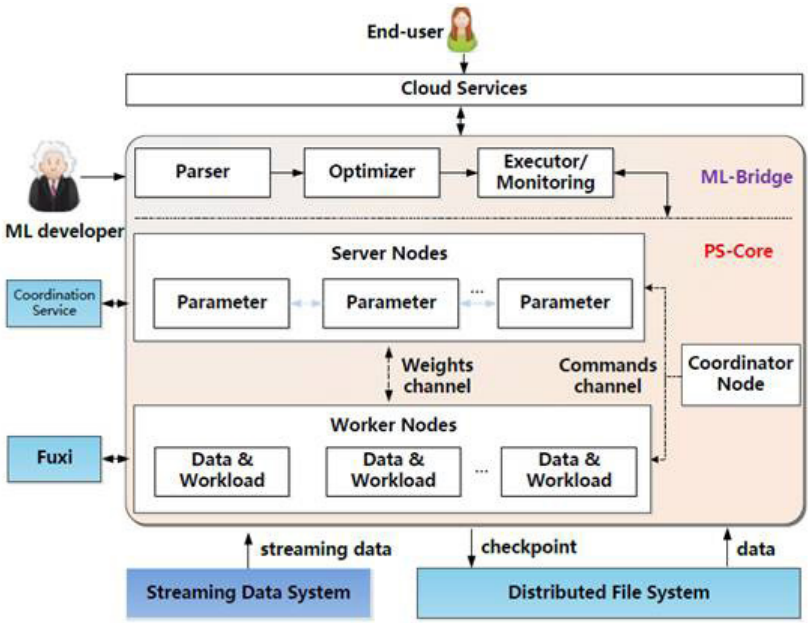


图 3　鲲鹏的架构

鲲鹏的架构如图 3 所示，它建立在阿里巴巴集团内部的大规模分布式 Apasra 平台上面，拥有 Robust Failover、Backup Instance，以及 DGA for Scheduling &Synchronization 等特性。图 3 中的核心模块包括以下几部分：

Server nodes：对模型做分片存储

Worker nodes：对训练数据做分片并计算

Coordinator：控制算法整体流程，如初始化，迭代，终止等

ML Bridge：使用脚本形式的工作流对数据进行预处理

PS-Core：核心的参数服务器组件 (servers/workers/coordinator)

Fuxi：监控所有机器运行状态，必要时进行容错

用户视角

```
    --data synchronization
    sync xxSrcData to xxDestTable;
    fea_engineering -i xxDestTable -o feature_engineering_result;
    --prepare training data
    create table if not exists demo_batch_input (labels string,kvs
string);
    insert overwrite table demo_batch_input  select labels,kvs
    from feature_engineering_result where xxCondition;
    --KunPeng training
    ps_train -i demo_batch_input -o demo_batch_result -a xxAlgo -t
xxTermination;
    --deployment
    run prediction and evaluation
    copy demo_batch_result to xxSystem;
    --other workflow
    AB Testing
```

图 4　鲲鹏架构用户视角

鲲鹏系统的调用，对普通用户而言也非常简单。用户只需要使用简单的几行脚本形式的命令，即可完成整个算法的调度。整个过程主要包括：

1. 数据预处理，准备成算法接受格式
2. 构建算法的输入 / 出表
3. **调用鲲鹏算法**，ps_train –i demo_batch_input –o demo_batch_result –a xxAlgo –t
4. xxTermination;
5. 评估算法效果
6. 进行 A/B 测试

从图 4 中可以看出，整个流程对用户而言都是透明的，使用过程也“如丝般顺滑”，不用感知算法背后复杂的优化及调度过程。

开发者视角

```
ParameterVector mServerParam, mServerGrad;
DataVector mWorkerParam, mWorkerGrad;
tr1::shared_ptr<SubDataset> mSubDatasetPtr;
// core code
bool Iterate()
{
  mServerGrad.Reset(0);
  // how many steps before an all-worker sync
  for (int i = 0; i < mConf.nSync; ++i)
  {
    // determine if need to pull parameter from server
    if (0 == i % mConf.nPull)
    {
      mWorkerParam = mServerParam;
      mWorkerGrad.Reset(0);
    }
    DatasetCal("CalculateGrad", *mSubDatasetPtr,
      CalcNDArrayHelper().AddNDArray(mWorkerPara
      m).AddNDArray(mWorkerGrad));

    // determine if need to push grad from server
    if (0 == i % mConf.nPush)
    {
      mWorkerGrad.ReduceTo(mServerGrad,
       "UpdateParam",
       CalcNDArrayHelper().AddNDArray(mServerGrad
       ).AddNDArray(mServerParam) );
    }// end if
  }// end for
  SyncBarrier(); // do synchronization
  // other codes
}// end Iterate()
```

图 5　鲲鹏架构开发者视角

鲲鹏架构对普通的机器学习算法开发者而言也非常简单。它将复杂的通信及调度过程包装成了 API。如，Worker.PullFrom(Server)，开发者只需要这一行简单的代码即可把模型从 server 端 pull 到 worker 端。再如，SyncBarrier()，这开发者只需要这一行简单的代码即可完成 server 端模型的同步。

实验结果

与 Spark 和 MPI 的比较

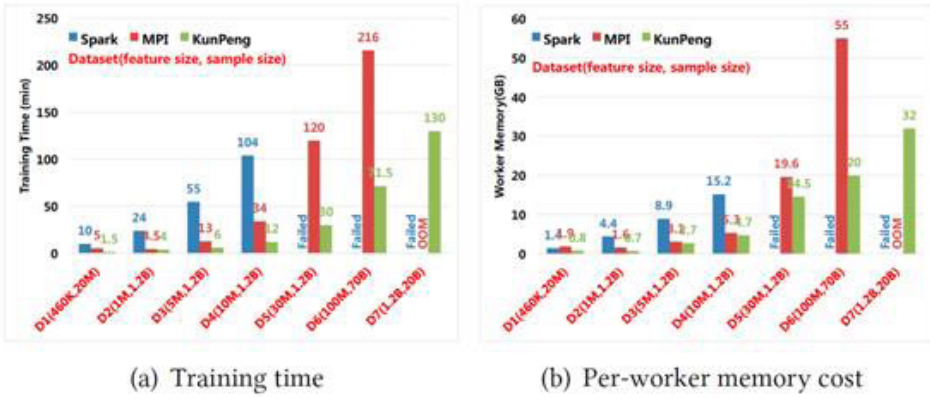


图 6　鲲鹏与 Spark 和 MPI 训练时间及内存消耗对比

图 6 显示了在七个不同数据集上（D1–D7），鲲鹏与 Spark 和 MPI 的逻辑回归算法（LR）训练时间及内存消耗对比。如 D1(460K,20M) 指该数据集包含了 46 万特征，2000 万样本。从中可以看出，Spark 和 MPI 的 LR 在特征超大的情况下（D7）会出错，而鲲鹏的 LR 则可顺利训练成功。

Kunpeng-MART 与 XGBoost 比较

Dataset	Description	#Features	#Sample number (million)	KunPeng-MART	XGBoost
Rec CTR	Items recommendation for sale	78	84	162G	176G
Search CTR	Search results ranking	592	1,000	1,775G	1,810G
Ads CVR1	Search ads ranking	698	2,067	3,711G	3,912G
Ads CVR2	Search ads ranking	698	10,245	22,157G	N/A

图 7　Kunpeng-MART 与 XGBoost 内存消耗对比结果

图 7 显示了基于鲲鹏实现的 MultipleAdditive Regression Trees（MART）与开源的 XGBoost 在四个不同数据集上的对比结果。从中可以看出，基于鲲鹏的 MART 内存使用情况要稳定的低于 XGBoost。此外，我们在 Ads CVR2 数据上重复跑了 10 次 XGBoost，但无一成功得到结果。图 8 显示了基于鲲鹏的 MART 和

XGBoost 在相同数据集上运行时间的对比，其中也可以看出基于鲲鹏的 MART 训练时间要优于 XGBoost。

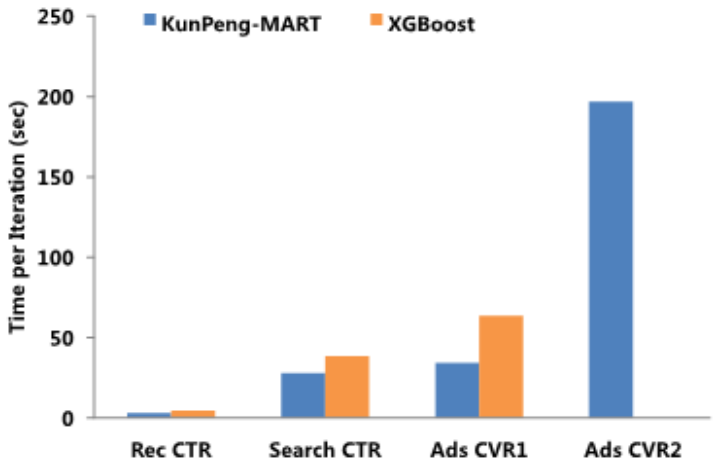


图 8　Kunpeng-MART 与 XGBoost 训练时长对比结果

Worker 数量对算法的影响实验

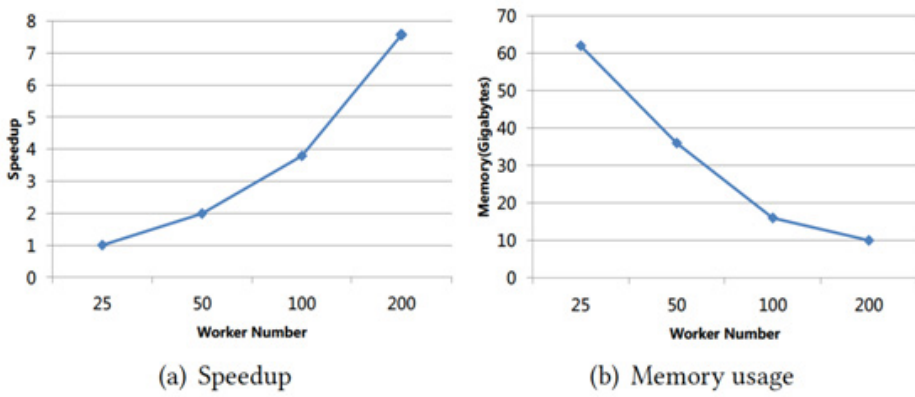


图 9　Worker 数量与算法加速及单 Worker 内存使用关系

图 9 显示了 Worker 数量与算法加速及单 Worker 内存使用的关系。在该实验中，我们使用的是基于鲲鹏的稀疏 LR 算法，特征约有 70 亿个，样本约有 180 亿个。从中可以看出，25 个 worker 就能训练这些数据。而且随着 worker 的增多，算

法训练速度倍增，同时单机上的内存使用会倍降。

总结

本文所提出的分布式学习系统——鲲鹏，拥有强大的分布式计算能力和算法优化能力，同时也有用户友好的界面和接口。在实际的在线或离线任务中，它能接受百亿特征，千亿样本和万亿参数。同时，它在生产集群中，有着很好的健壮性、灵活性、扩展性及高效性。此外，它在阿里和蚂蚁众多实际场景中发挥出了巨大的优势。例如，在 2015 年“双 11”中，鲲鹏系统上实现的“楼层”排序（LR 算法）使得 UV CTR 提升了 21%，GMV 提升了 10%。

再如，基于鲲鹏实现的 GBDT+DNN 算法应用在支付宝交易风险评估业务中，该算法上线以来，相同覆盖度的情况下，案件召回率从 91% 增加到 98%，每天减少了几千万次用户的打扰。此外，在鲲鹏上实现的 Deep Structured Semantic Model（DSSM）模型，已经广泛被应用于神马搜索，淘宝搜索，1688 广告，蚂蚁智能客服等业务中。

总体来说，鲲鹏系统上的 10+ 个成熟算法已经被广泛应用于 120+ 个产品中，这些无一不是阿里生态体系内最大规模的算法。

团队：蚂蚁金服人工智能部 & 阿里云

作者：周俊，李小龙，赵沛霖，陈超超，李龙飞，杨新星，崔卿，余晋，陈绪，丁轶，漆远

大数据

深度 | 两个案例，掌握 AI 在大数据领域的前沿应用

鸿侠

阿里妹导读：近日，全球技术学习技术大会首次在京举行，阿里巴巴数据技术及产品部资深算法专家杨红霞（鸿侠）作为特邀嘉宾出席并发表主题演讲。鸿侠从什么是数据新能源说起，接着介绍了阿里目前比较成功的两款数据产品，一个是是自动化标签生产，另外一个是大规模分布式知识图谱，以及在此之上的一些重要应用。最后是她对机器学习和人工智能技术对数据新能源产业中有效落地的一些建议和期望。

下面是基于鸿侠现场演讲内容摘要：

如果大家对阿里巴巴的新闻比较关注，最近可能会频繁听到阿里巴巴谈到“五新”这个词，“五新”中的其中一个概念是新能源。**其实新能源就是大数据本身。**技术、数据和算法三个方面结合在一起，才可以把数据真正用起来。

大家都知道，Google 的数据量很大，但是它的数据源本身其实比较单一。以 Google search，Google map 等为主导。再来看看 Facebook，它更多的是社交行为数据，缺少出行数据、浏览器数据、或者类似优酷的视听数据。但是，对于阿里来说，上述的这些数据我们都有。我们面临的极大挑战是：怎么样有效的把这些全域数据融合在一起。

首先我们需要把数据有效地收集起来。把数据有效地收集、存储起来之后，接着要做的就是怎么通过算法把这些数据打通，并且真正有效、智能地把这些数据提炼出来。

这是阿里的一个生态体系图。最底层是阿里云，这是我们的一个计算存储框架。上面是阿里妈妈，阿里妈妈是负责整个阿里巴巴计算广告的一个部门，再上面是菜鸟、支付宝和蚂蚁金服。然后是与电商业务相关的，像淘宝网、天猫、聚划算等等，或者是跟文娱相关的，优酷土豆，还有像阿里旅行，口碑之类的业态。

阿里巴巴数据中台要做的事情是什么呢？举一个最简单的例子，之前有一个比较火的电视剧《三生三世》。《三生三世》火热上映的时候，与之相关的商品元素，比如饮食或者穿戴之类的商品，也会瞬间在淘宝网上火爆起来。那么如果我提前就知道某一类人群是《三生三世》的粉丝，我就可以在淘宝网上做非常高效的、准确的定位推广。阿里数据要做的是：把数据真正打通，深度挖掘数据的价值，为业务创新应用提供数据决策基础和依据。

下面具体介绍一下数据融合的技术框架。因为在真正进入算法之前，我们一定要对数据进行非常认真、仔细地进行清洗过程。俗话说，如果你的数据不清洗，其实就是“learn trash from trash”。所以数据本身一定要做得非常干净。

首先来看一下架构图，第一个数据层中有各种各样的数据，比如有消费数据，有广告数据，出行数据等等。把这些数据层经过有效结合在一起之后，接下来得到这种特征层的提取。在阿里数据内部，大概有这样几个比较抽象的维度：像账号设立的静

态特征，电商行为的特征，或者设备的特征等等。

在特征层之上，我们会有模型层，这里面有基于业务规则的模型，也有其他的例如异常检测，有监督或者无监督的学习，然后特征的联合校验等模型。因为我们的数据源非常多，因此我们也可以通过部分的数据源验证另外一个数据源，看数据的增长或者留存是否处于一个正常范围。另外还有一些比较好的方法，比如基于 Graph 的一些算法，实时的反作弊算法等等。在算法层之上，就是评估层。在评估层内，我们可以判断留下来的数据是否是真正有效的数据。

在上述这些数据层的上面，会有一个应用层，也同时会抽象出一些产品来帮助内部员工或者外部商家进行使用。所以，整个数据中台实际上是从底到上对数据进行清洗的一个架构。

当我们有了非常干净的数据之后，我们要做的就是把数据打通。我刚才说了，阿里生态体系会呈现出几百个不同的数据源，这些数据源本身的数据量非常大，收集模式也各不相同。那么我们是如何进行数据之间的融通的呢?

上图是这是我们关于怎么把数据打通的一个技术架构。大家可以看到，整个技术体系都是，先把数据接进来，再通过一些机器学习或者深度学习的办法（像 word2ve，node2vec，TFIDF，归一化等）处理特征层，之后映射到一些比较抽

象的高纬度 Level（比方说像用户的身份信息，网络的环境相似度，文本的相似度，APP 相似度等等）。抽象完这些特征层之后，我们究竟怎样去判断。

这期间的方法大致可以分为四种有效的办法：

- 深度学习的模型
- 非线性模型
- 线性模型
- 图模型

此外，还有一些基于规则的强召回，就是比如说用户有相同的账号登陆不同的地方。这些是所谓的强召回，它可以非常准确地被判断出来。弱召回就是基于算法特征层的这些模型，有效地判断出所有信息是否真正属于同一个自然人。下面，基于刚才的打通融合的数据之后，介绍两个数据应用类产品。

1. 自动化标签的生产

在电商业务中，我们想知道这个人背后更多的其他方面的特征。所以，我们会有一个自动化标签生产的体系，快速地进行标签生产。比方说上传一些种子用户，比较类似于像 Facebook 的 Look alike audience，可以快速在几千万个特征中选出来最重要的那些特征，然后通过最重要的这些特征对于那些还没有打标的庞大人群进行打标。

在阿里数据中台，我们研发了一个叫“自动化标签生产”的体系。这个体系需要满足三个需求：

需求的响应速度一定要快。

标签生产的负荷能力要强。

对于这个标签生产数据源是有一定要求的，就是你要做出判断，不是他上传了一批种子用户，他对某一些标签有需求，我们的数据量，或者数据就一定是足够帮你产生这些标签的。

所以，基于刚才的一些要求，我们推出了“标签工厂”的这一套服务体系。它可以达到几个目的：

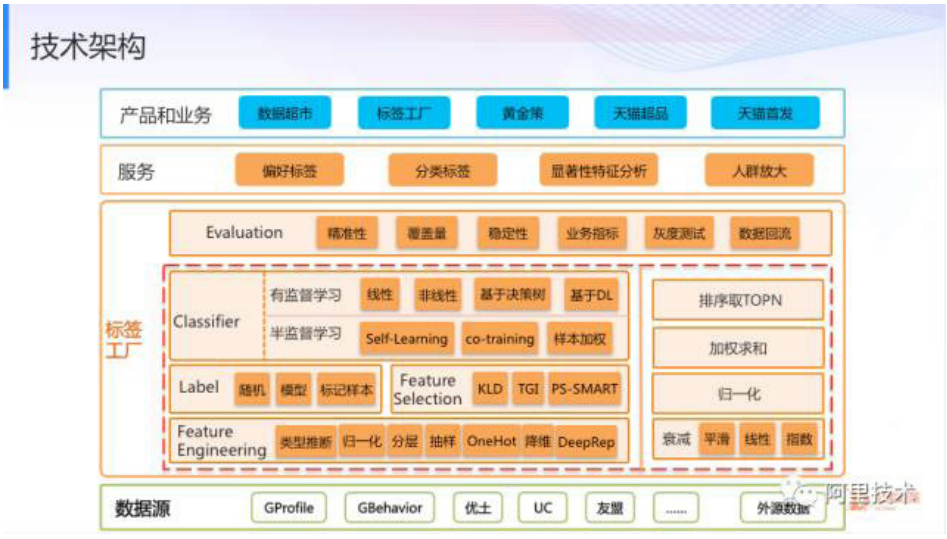
降低成本。现在我们有一个可视化界面，只要你上传一个种子用户，按照你自己

的要求，大概在一两个小时之内，帮你快速生产标签。

质量。当你在产生第一轮标签的时候，其实更多的还是基于对算法本身的一个评估判断。标签真正上线之后，在业务的指标上面会不停产生反馈，所以我们实时的把这些业务反馈放进在标签生产体系，不停地帮助优化标签的产生。

保证数据安全。

下面具体看一下我们的技术架构：



第一，数据源。你可以认为，整个数据新能源的数据源都是接到下面，经过数据清洗、打通之后，来到一个标签工厂的体系。在标签工厂，首先会进行一定特征学习（feature engineering），比方说有一些像类型判断、分层、降维，因为数据量非常大，通过深度学习，深度表征去学习出特征之间的非线性关系，和它们之间的 high order interaction。

接下来就是打标。首先可以传一批种子用户，由于你打标签的这部分种子用户是非常小的一部分用户，所以还涉及到快速扩充 Label，或者通过不停的这种 adaptive learning 去训练完之后，可能通过算法的输出，可以增加一些更有效的确实是能反映出你的 Label 真实的样本集。

如果你的标准样本很多，很丰富的情况下，你可以用有监督的学习。有监督

的学习，其实有线性的、非线性的或者是基于深度学习的。真实情况是有 label 的 sample 很少，在更多情况下我们需要使用半监督的算法，例如 self-training, co-training 等。

2. 大规模分布式知识图谱

讲完标签工厂之后，介绍另外一个产品，大规模分布式知识图谱。大规模知识图谱抽象也是一种图计算。首先谈一下基于大规模分布式知识图谱做了哪些工作，以及我们为什么要做这样一件事情。

阿里巴巴的生态非常丰富，而丰富的业态背后给我们数据工作者带来的困难就是，我们常常需要接入各种数据，并将他们有效地管理和整合起来，传统的方法，我们可能需要花几个月，投入几十个人做这样一件事情，对数据进行打标。

但是，假设我们已经知道数据和数据之间的一些关系，而且也知道数据表中哪些表之间调用的血缘关系。那么，如果我只是对调用次数最多的表进行非常精确地打标，然后用基于知识图谱的方法，对剩下的 90% 表进行推理式的 Label 打标，就能极大节约了人力成本。所以这就是我们为什么要用知识图谱去做数据接入这样的事情。那么，对于数据管理也是同样的道理。

假如只有 1G 的数据，你可以很快地回答出数据分布的情况和质量。而我们的现状时，我们的数据达到 ZB 级别的规模。因此对数据管理来说，挑战不容小觑。同样地来看看数据应用方面的情况。我们基于数据应用，实际上也有一款产品叫做“数据地图”。

数据地图是干什么呢？其实就是当你进行一个查询，在这个产品里会自动帮你反馈出一个最相关的表。延伸开来的是我们想要做得下一步工作：当你下一次进行查询后，能直接返回出相应的 SQL，再产生出相应的表和相应的结果——这是我们想做的智能取数。

下面介绍一下知识图谱在数据管理和数据应用方面的落地进展。目前我们开发了一些基于几十万张、上百万张 ODPS 表的知识图谱。说一下我们的结果：

在数据资产管理中，有一项重要的工作就是判断数据的归属。我们有上百万张的

线上表，其中可能有几万张到几十万张的表能够比较清楚判断是属于哪个团队，可以在数据版图上打标。但是，还有上百万张没有打标的表，因为这个表属于异构的。在之前，通过一些人工的规则，它的归属判断准确率大概是 55%，而通过前面介绍的知识图谱框架，准确率可以提升到 88%。所以，它对数据资产管理的准确性起了显著的提高作用。

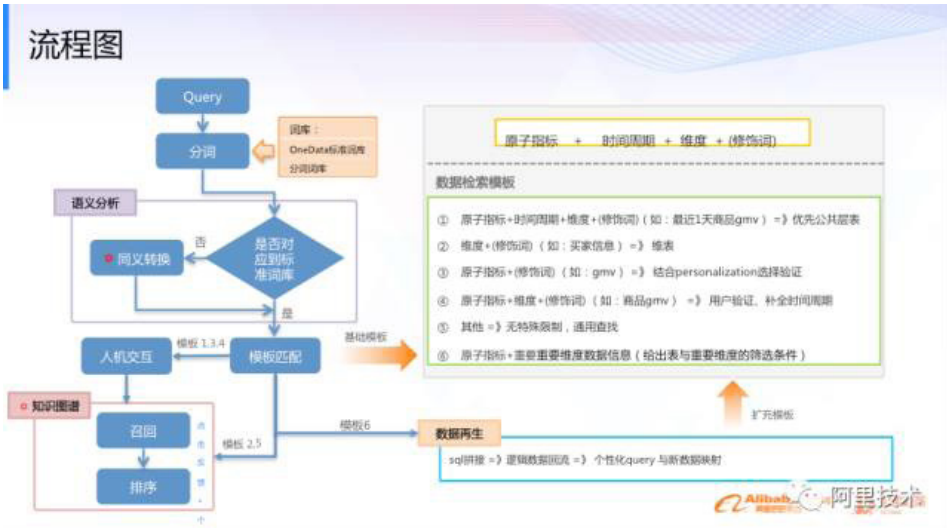
接下来讲一下知识图谱在数据应用当中的一个技术框架，其实也是比较类似的：

首先，数据层。因为是一个知识图谱的构建，所以上面要加个辞典层和语义层。再上面就是基于推理层。在推理上，用的方法有大家比较熟悉的像随机游走和延伸等。那基于标注的，我们其实尝试了很多种方法，比方说张量分解等。

目前为止比较成功的是 PRA(path rankingalgo)，我们研发了几个主流的 graph feature model，PRA 在大规模分布式知识图谱推理上，在我们的问题中，表现是最好的。什么是 PRA，其实是把这些路径抽象出来，然后就是学习一下再推荐这个路径，但是它对于我刚才说的很多文本信息并没有有效的利用起来，比如对于这些表的描述，在最原始的 PRA 当中路径本身的位置是有考虑进去的，当然我对于这些描述，可能会知道也许这个路径更有效。所以，后来我们看了一下这个 Trans 系列，其实类似 text analysis 中的 word2vec vs tfidf。确实在我们整个的刚才说的案

例当中也是有比较大的提高。

看一个具体的例子，在数据地图当中，知识图谱到底是怎么工作的：



你打出一个查询，首先就是基本的分词与分析，其实大家可以看到，我们这个场景也是相当于搜索反馈一个结果，但是其实它和传统的搜索是不一样的：传统的搜索像 Google、百度，其实它关心的指标是你准确的那个值是不是在 TOP5 或者 TOP10。

但是，我们这个场合下一定要反馈唯一的、准确的表。所以，接下来我们会有一些模板匹配，所以这里非常重要的一部分是人机交互这一块，把人机交互的结果，就是人要告诉你说，这个结果是否是他想到的，然后知识图谱整个刚才的框架有效的结合在一起，然后产生出你真正想找到那张表，然后整个的这个过程，其实这些是一些抽象出来的模板，这些模板可能是不够的。因为随着人的查询越来越多，模板也需要慢慢的扩展。现在我们还是基于一些规则判断一些模板，未来我们也会尝试，让这个机器自动产生一些模板。

所以总结一下，我刚才给大家大概介绍了一下什么叫数据新能源，以及我们在数据新能源上两个成功产品，一个是自动化标签的生产，可以在非常快速的在几个小时之内，为几亿人打上有效的标签，并且快速的验证落地。另外一个是大规模分布式

知识图谱，以及两个应用的比较好的产品，一个是数据资产管理，另外一个是数据地图，就是快速的查询这个有效的表。未来我们想做的不只是一个表本身，也许就是一个 query 对应的 sql，对应的你最后的结果。

最后再讲一下我们对于整个工业界中机器学习怎么才能成功落地的一些建议：

第一，必须要有大数据。如果没有大数据，其实很多的挑战你是看不到的。而且大数据本身一定是要丰富多样的，如果数据源本身过于单一，其实对模型的挑战是比较小的。所以你的数据源本身多样性比较要多，机器学习才会发挥出更大的作用；

第二，一定要有计算平台。像现在阿里云给我们提供了一个非常好的保障；

第三，你开发的算法一定是要通用的。就是大家可能是在这个公司工作会发现，每开发一套算法，投入的人力和时间和成本都是比较高的。所以说你的算法本身可延展性一定是要比较好的。

我今天的演讲就到这里。谢谢大家！

近 300 位数据挖掘专家云集阿里，最精彩的发言都在这儿

阿里技术

阿里妹导读：2017 年 6 月 29 日，中国杭州阿里巴巴西溪园区，首届数据挖掘前沿发展与未来论坛成功举办。作为阿里巴巴集团、中国中文信息学会和 KDD China 三方联合打造的国内业界和学界顶尖数据挖掘论坛，会议吸引了来自国内顶级高校和知名企业的近 300 名专家学者到场参会。

众多数据挖掘领域大咖如：阿里巴巴 iDST 负责人金榕、蚂蚁金服人工智能部技术总监李小龙、IEEE Fellow、ACM Fellow、AAAI Fellow 国立台湾大学教授林智仁、清华大学计算机系副教授崔鹏、中科院副研究员罗平等齐聚一堂，共同探讨数据挖掘领域前沿研究。

下面随阿里妹一起，看看诸位大牛的精彩观点吧！

淘宝“问大家”，实现大规模在线精准匹配

金榕，阿里巴巴 iDST（Institute of Data Science&Technologies 数据科学与技术研究院）负责人；美国密歇根州立大学终身教授，曾担任 NIPS、SIGIR 等顶级国际会议领域主席和 KDD、AAAI、IJCAI 等顶级会议高级程序委员会委员；ACM 中国理事会常务理事。

精准匹配的目标就是试着在作用者和任务间做出最佳的任务分配。每当你分配特定任务时，即某个作用者所要实现的目标，你将得到一个不同的参数。难点就是该如何发现最佳的任务分配，从而使整体的奖励参数是互补的。很多情况下，每个任务都只能被分配给有限的作用者，或者某个作用者只被允许完成少量的任务。

淘宝“问大家”功能实现了大规模在线精准匹配，比如有些人想要购买毛衣，但他对此持有疑问，系统发现确实有机会或有潜力回答这些问题的用户后，在抽象意义上匹配这个问题，从而对问题进行精准分配。“问大家”功能可以实现这样的效果，由于阿里一方面拥有在做出最终决定之前持有疑问的不同用户所提出的问题；另一方面拥有那些有潜力的回答问题的用户数据，这样系统就可以顺利进行在线大规模精准匹配。虽然如今大多数人聚焦于学习预测，但实际上，这是一个很长的过程。 我认为

未来以及接下来要做的事就是充分使用大规模在线匹配预测来为任务进行最佳安排。

林智仁：单机和分布式设置对于大数据机器学习都很重要

国立台湾大学教授、IEEE Fellow、ACM Fellow、AAAI Fellow。

在大数据时代，越来越大规模的数据需要处理。数据通常太大无法存储在一个电脑中，但何時该使用分布式机器学习是个仍需探讨的课题。采样数据存储在一个电脑是一个容易和直接的选项。而且根据过去的统计，一个观点是说电脑存储增大的速度比数据增大的速度还要来的快。但另一方面，因为互联网公司的数据已经存储在分布式系统中，如果我们直接进行分布式机器学习，工作流将不会中断。我认为传统的单机设置和新的分布式设置对于大数据机器学习都很重要，但实际使用遇到的问题决定采取哪个途径。

业界、学界思想精彩碰撞

图中从左至右

鸿侠（主持人）：阿里巴巴资深算法专家

盖坤：阿里巴巴资深算法专家

李小龙：蚂蚁金服人工智能部技术总监

崔鹏：清华大学计算机系副教授

罗平：中科院计算技术研究所博士生导师、副研究员

杨洋：浙江大学计算机学院讲师

鸿侠：各位对机器学习、数据挖掘、深度学习这几个是如何理解的?

盖坤：首先说机器学习和数据挖掘。数据挖掘是一套完整的理论，根据自己的问题从数据里挖出有用的信息，用简单的统计方式或者用各种各样的方式都可以。机器学习是一套方法论，用在数据挖掘的问题里，也可以用在视觉、语音等等其他问题里面。

从现在的发展来讲，数据挖掘里面比较复杂的问题和前沿的问题，很多都和机器学习有关，这个关联或多或少。其实如果你真的在数据挖掘领域里面想要做的不管是

深度也好，还是广度也好，做的比较好的话机器学习应该是必备的技能，虽然不是逻辑对等的相关关系。

再说深度学习和机器学习。现在深度学习基本上已经快占了机器学习的 80% 的概念和资源了，我做研究生的时候大家还不太认可深度学习，现在大家也在慢慢转变观念。近几年，深度学习展现出来其效果和复杂性，慢慢在各种领域开始展露非常强劲的性能，解决实际问题的威力比较大，所以在机器学习里面越来越重，工业界的资源也有很大的投入。

从研究角度来讲，以前做机器学习问题，从问题定义到数学定义再到求解整个一套方法其实是研究人员需要全套推导的，首先问题是什么，有了问题可能转化成一个优化问题再研究优化方法，优化问题能不能解很关键，模型很好不能解也不行。工作很重，模型设计也有制约。

我看到深度学习展现出来几个优势：

第一个，优化方法标准化，研究工作量大大降低，解开束缚。优化方法就用 BP 以及 BP 衍生类方法，有一套标准化方法，使得做模型的人不用那么关注优化方法了，虽然必要的理论认知和调试工作必须要做，但很多时候优化问题不再是一个拦路虎了，基本上是通的。

第二个，模型组件化，可以构建更复杂模型。之前从头到尾构建机器学习方法的时候，像现在深度模型的复杂度是可望不可及的。现在优化方法解耦之后，又在模型上面变成组件化。可以用现在基本单元，有 LSTM 等基本单元，也可以自己创造基本单元，可以在里面发挥创造力，比如新创连接函数，层数也没有太多束缚，可以组件化地构建一个非常复杂的模型。

第三个，深度的方式。虽然没有理论证明，但我认为在泛化性能上也有一定优势。理论上，潜层模型，像非常多隐层节点的单隐层网络，近邻法或者普通 Kernel 方法，也是有无限复杂的拟合能力的。但潜层网络更像是记忆器，只记住训练样本，会过拟合，泛化性不是很好。而深层网络，如果设计模型结构更匹配这个数据，例如深层 CNN 在图像上，实践中泛化能力上是不错的。

李小龙：这里我补充一下，从工业界来讲，深度学习现在可能像是黑洞，把所有

机器学习的注意力都吸到里面去了，这个有好也有不好的地方。对于深度学习还是要有清醒认识，它有一些难以克服的问题：

一个是不可解释性。这就造成到底是哪个原因导致你这个模型效果好，是不知道的，比如像金融场景有一些对解释性要求非常高的，比如风险模型，为什么不给这个人贷款，用深度学习就没法做到；还有芝麻信用分，为什么他的高，为什么他的低，如果用深度学习来做，也是没法解释清楚的。这个也是实际应用中必须认识到深度学习还是有缺陷的。

第二个深度学习对数据的要求非常高。深度学习网络复杂度高，需要大量的数据，也导致了它很难在常规的小数据的场景下能够起到很好的作用。事实上，现在学术界对小数据已经开展研究，比如只需要一个样本就能够建出来较好的模型，这也是一个值得关注的方向。

鸿侠：深度学习这么火，但很多人在质疑深度学习虽然可以做到很深几十层、几百层，还有上千层但解释性却很差，但是可解释性强并且有理论保证的 Hierarchical Bayesian Model 却并没有火起来。各位怎么看?

崔鹏：我觉得深度学习比较吸引人的地方，就是比较标准化，明白输入输出参数的意义就可以进行研究，这个对于工业界来讲是一个好事情。但对深度学习持比较保守观点是我从教育学生的角度来思考的：我认为做学术研究应该是一步一步在走的，原来像 SVM 研究兴旺很多年，现在又转到深度学习上面，作为一个研究人员，我觉得最好保护自己的方式就是要用一个相对来说比较完整或者成体系的理论武装自己。

深度学习之外其他的一些前沿方法，之所以没有在产业界推广开，一方面可能的原因是理论门槛过高，怎么在可控的条件内去调整它，不容易标准化；另一方面是技术发展到一定程度，比如说刷榜刷到顶了，就需要具备其他理论基础的人来进一步研究。所以，从研究的角度，我认为越火的东西越值得我们谨慎看待。

罗平：对此，我也做了一些分析，比如语句稍微改一下，模型不知道怎么回事就不能正确反应了。也就是说，深度学习模型的鲁棒性并不好。前段时间有篇文章也写道这个问题：自动驾驶很多时候都会驾驶的很好，但如果从摄像头输入一些“人工扭曲”过的影响，可能会把车带沟里去。这就引申出 model testing 和 model verification 的

问题。深度学习是一个很强大的拟合工具，但还不是一个强大的泛化工具。

鸿侠：关于研究资源利用，在工业界大家有丰富的数据和计算平台是否可以介绍一两个基于此的成功案例？对于学术界，有的时候缺乏数据计算平台又缺失，这是不是个很大的挑战？

罗平：我之前是在工业界的又转到学术界。我觉得学术研究有两种模式：一种是从数据出发发现问题，抽象问题。因为计算机本质上应用型科学，只有解决实际问题的研究才是好的研究。另一种是从模型出发，做出一些“惊世骇俗”的工作（例如 PLSA 和 CRF 的发明）。

但如果仅仅是模型上的修修补补，其实研究价值并不大，因此，一切的基础都是建立在数据之上。工业界有大量的数据，学术界确实需要与工业界深度合作；而学术界也不能只关注工业界当下的问题，必须有前瞻性和开拓性。

关于数据的问题，还有一种模式是，找一个好的切入点，大公司可能不太关注这个点，我们自己来标注一些数据，这个数据的产生也是对学术界的一个贡献。我们现在也是试图朝着这个方面努力。但不管怎样，数据都是我们的基础。

杨洋：我想起了我朋友打过一个比方，说我们是干什么的，我们更多是摸索道路，以前没有人走过，有可能通向世外桃源，有可能哪也去不了，走到底了，告诉其他人可以通往哪里，这是我们学术界在做的一件事情。对于工业界而言希望这条路通往世外桃源，走了一年发现走不了，老板们肯定不会开心，到了世外桃源以后能干什么，这里有很美丽的风景，更重要的要开垦这片新的土地，让更多的人可以来这个新地方居住起来，在开垦的过程中就不是我们学术界的资源和能力独立完成的事情，这个时候工业界人会进来。

五六年前我们跟公司做一些合作，那个时候对于同一批数据，我们实验室对这批数据的想法和公司的一些想法是有很大的差异，公司想到是做推荐算法，我们实验室就想做一些其他更好玩的事，那个时候我们没法劝说公司直接做这个问题，我们会先赢得公司的信任。现在和工业界合作，比如做推荐或者 CTR 预估，很多公司有非常强大的团队可以做的非常好，比我们学术界好很多，公司很多情况下也会往长远想一步，合作就可以研究一些很好玩的问题。

权威详解 | 阿里新一代实时计算引擎 Blink，每秒支持数十亿次计算

王峰

作者介绍

王峰，淘宝花名”莫问 "，2006 年毕业后即加入阿里巴巴集团，长期从事搜索和大数据基础技术研发工作，目前在计算平台事业部，负责实时计算北京研发团队。

在阿里巴巴的 11 年工作期间，持续专注大数据计算与存储技术领域，基于 Hadoop 开源生态打造的数据基础设施一直服务于搜索、推荐等阿里核心电商业务场景，最近一年带领团队对 Apache Flink 进行了大量架构改进、功能完善和性能提升，打造出了阿里新一代实时计算引擎：Blink。目前数千台规模的 Blink 生产集群已经开始在线支持搜索、推荐、广告、蚂蚁金服等核心实时业务场景。

王峰在清华大学演讲

实时计算时代来临

随着互联网应用的普及、智能硬件的发展，数据的种类和规模都呈现了爆炸式的增长，各行各业都希望能够从大数据中发掘出更深层次的信息和知识，并产生实际价值。数据挖掘手段也逐渐从基本的数据统计向更高层次的机器学习和深度学习演变，但这些都需要强大的计算能力作为支撑，因此大数据价值的体现离不开大数据计算平台的发展。

目前大数据业界在计算技术上已经取得了显著的成果，例如：第一代开源的大数据处理技术 Hadoop 已经可以处理超大规模的数据集合，第二代开源的大数据处理技术 Spark 更好的利用了内存，并进一步加快了大数据处理的性能。

各大公司也都基于自身业务场景和数据规模定制了自己的大数据计算平台，但这些大数据计算平台大都是批处理系统，虽然具备海量数据处理能力，但在时效性上有明显的滞后。显然，数据的价值不仅体现在空间维度上，同时也在时间维度上进行伸展，很多场景下数据的价值也会随着时间的流逝而逐渐消失。因此，大数据计算平台需要能够尽可能的提升计算的时效性，越快地从数据中挖掘出信息就意味着能够获取到更大的价值。

时效性对数据价值的影响尤其在电子商务领域更加明显。通常人们在不同时刻会有着不同的消费需求和潜在目标。很多时候，这些需求和目标都是临时的（即和历史行为关联度较低），并且从产生到结束之间的时间是非常有限的。这种情况在阿里巴巴双十一大促这样的场景中表现的尤为明显。

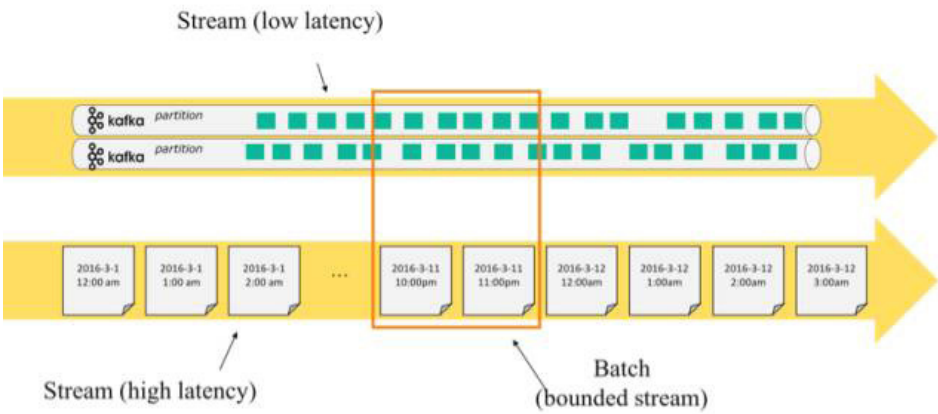
大促场景下，用户会由于丰富的促销活动和环境而临时产生更多的购物需求，并且每个购物需求的有效期是有限的。因此，搜索和推荐系统需要及时发现用户的需求变化，在数据有效期内完成模型更新，推荐用户当前感兴趣的商品。此外，阿里巴巴的数据大屏也需要在大促期间实时展示成交额等大家关注的统计信息，而不是大促结束后第二天再让大家看到数据。

其实目前不仅在阿里巴巴，各个行业都对大数据时效性的计算需求在日益增加，因此，阿里巴巴需要研发世界级一流的流式计算引擎，实时处理海量数据，提供在线统计、学习和预测能力，不仅支持阿里巴巴自己的核心电商场景，同时也能通过阿里

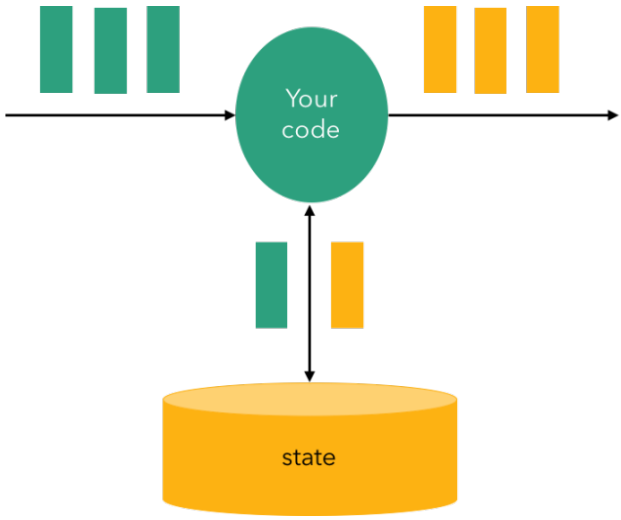
云向外部中小企业提供流式计算服务，输出实时计算能力，这就是我今天要分享的最新一代阿里巴巴实时计算引擎 Blink。

流式计算介绍

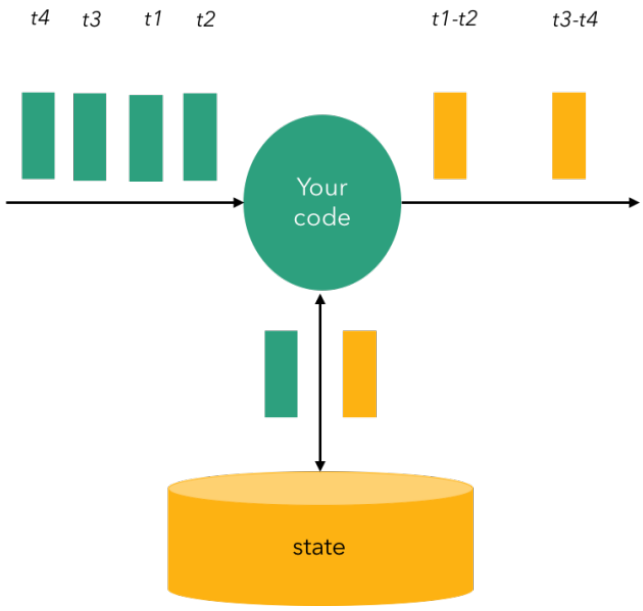
显然批量计算模型是无法满足当前大数据实时计算需求的，只有流式计算模型才是实时计算的天然计算模型，因此我先介绍下流式计算的基本思想，尤其是区别于传统批量计算的一些概念。批量计算是对于有限固定的数据集合进行处理，流式计算是对无限数据流的处理，即计算无法确定数据何时会结束。从另一个角度看，批量计算其实也可以认为是流式计算的一种特例，因此批量计算可以看做是一个数据流中的片段，即有明确开始和结束标记的数据流，如下图所示：



完善的流式计算不仅应该提供实时计算能力，还应该支持计算过程中的状态管理，状态主要是指计算过程中需要的数据或者变量，例如：统计计算中的 aggregation(sum/min/max…)，机器学习中的 feature 和 model，状态管理包括这些数据的存储、备份、恢复，版本管理，提供读写访问 API，并保证一致性，如下图所示：



此外，完善的流计算还需要考虑数据的时序问题，因为现实场景中，数据的产生顺序和接收顺序未必一致，因此需要给数据附带时间戳属性，即：event time，计算逻辑可以按照数据的 event time 来处理，这样可以解决数据的乱序问题，配合 watermark 机制，可以较好的解决 time window 计算，如下图所示：

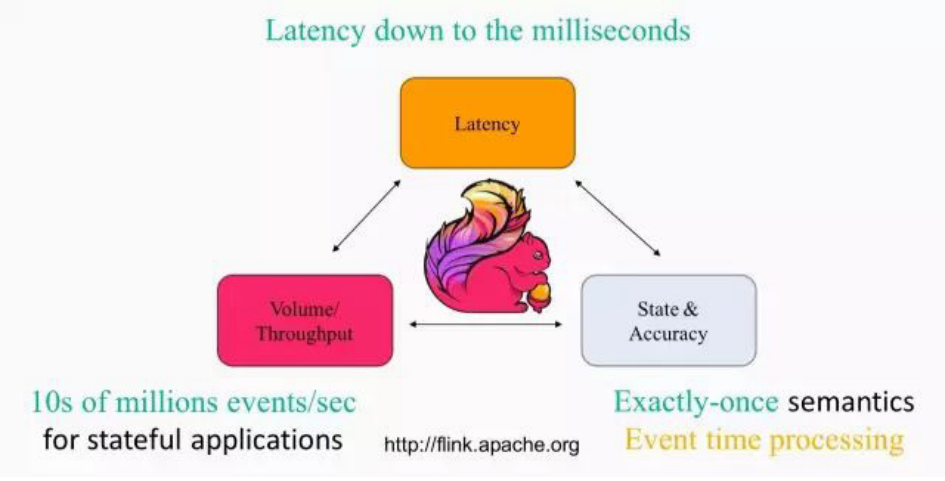


在开源大数据生态中，Storm 是第一代流式计算框架的代表，但它不支持状态管理，即 Storm 中的状态数据需要用户自己存储在外部存储系统中，数据的持久化和一致性都需要用户自己保证，这会给应用带来一定复杂度。

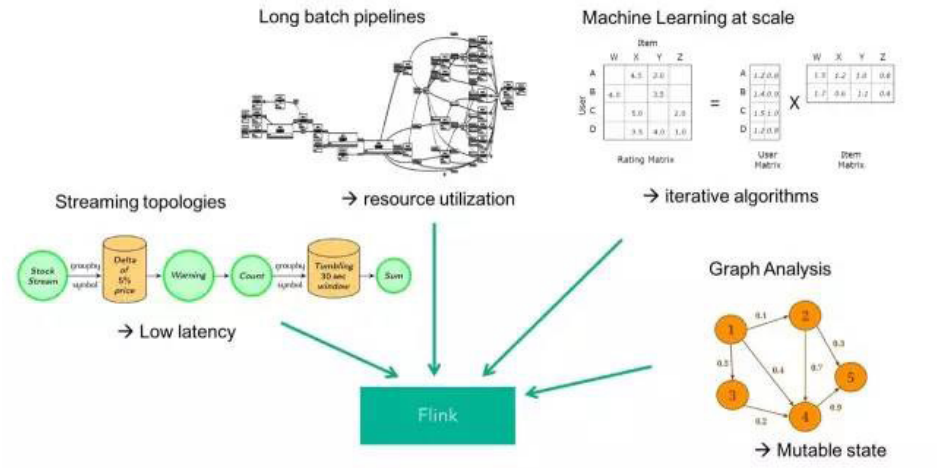
后续出现的 Samza 是支持状态管理的第二代流计算技术，内部利用 leveldb 和 kafka 来存储数据，但 samza 只能保证 at least once 不丢数据，但无法保证 exactly once 的强一致性，这在一些严格的场景下是有局限性的。近几年火爆的 Spark 也很快推出了配套的 Spark Streaming 技术，但其本质上是通过连续不间断的 Mini Batch 来实现流式处理的，不是纯粹的流式计算思想，时效性上也有一定局限性。

只有最新一代的 Flink 是相对最为纯粹和完善的流计算技术，在理论模型上具备了一切流计算的特质，也是支持 Apache Beam 最好的 Runner，给我们启动 Blink 项目带来了非常有价值的启发，因此下面我将介绍下 Flink 这个 Apache 开源流计算项目。

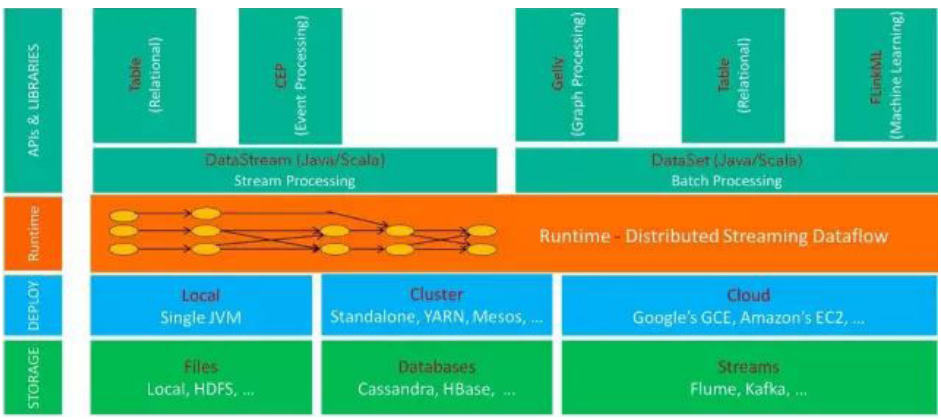
Flink 介绍



流和批统一的计算引擎

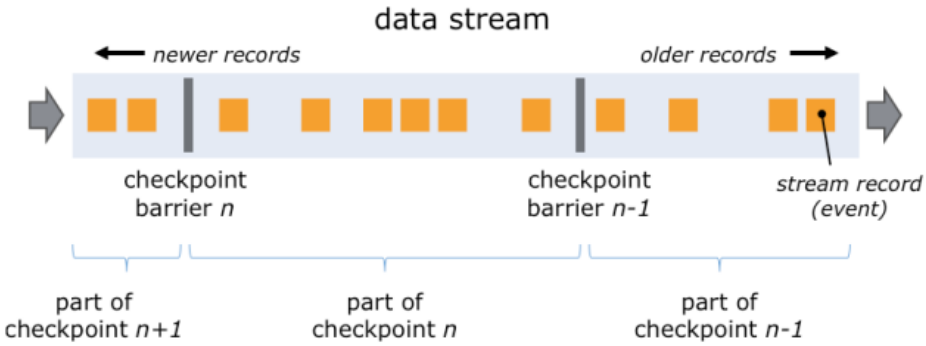


完整的生态系统

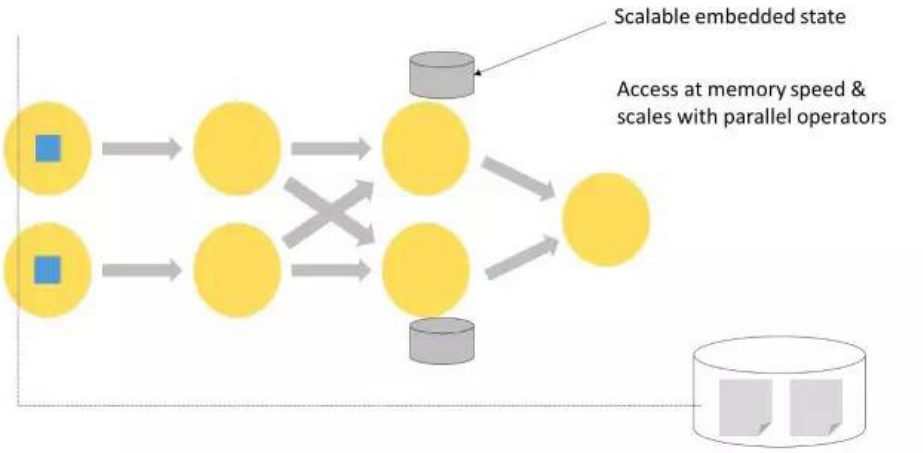


状态管理和一致性

Chandy-Lamport 算法是 Flink 支持状态管理和强一致性的核心理论基础，算法基础思想如下图所示：



Chandy- Lamport 算法的核心思想就是定期在流式计算任务中插入 Barrier，然后触发整个流做一次 Checkpoint，即将任务的 State 做一次 Snapshot 持久化保存。在下次任务重启的时候，可以基于上次成功的 Checkpoint 进行恢复，过程如下图所示：



Flink 的问题

综上所述，Flink 是一套理念和架构设计非常先进的流处理引擎，并几乎支持了流式计算所有的特质，但 Flink 发展尚在初期，在活跃度和成熟度上稍有欠缺，并且尚未在业内得到大规模生产实践的检验，因此是无法直接应用在阿里巴巴这种级别的生产场景中的，因此我们在 2015 年下半年启动了 Blink 项目，目标是扩展、优

化、完善 Flink，使其能够应用在阿里巴巴大规模实时计算场景，并将此项目命名为 Blink，下面我将介绍 Blink 的设计以及在阿里巴巴的应用。

Blink 介绍

Blink 产生背景

在 2015 年，当时我们还是阿里巴巴搜索事业部的数据技术团队，负责阿里巴巴所有商品搜索后台的数据处理，包括淘宝，天猫，B2B 等全球商品，面对海量商品的数据处理，我们需要在维护两套数据处理流程，一套是每天晚上的全量流程，同时还要一套白天的实时增量流程，为了降低开发和维护成本，我们开始探索一套流和批统一的计算引擎。

当时我们重点分析对比了 Spark 和 Flink 两套技术，最后虽然觉得 Spark 相对成熟稳定，但 Spark 是从 Batch 出发，模拟 Streaming，而 Flink 正好相反是从 Streaming 出发，认为 Batch 是 Streaming 的 Special Case，因此我们感觉 Flink 的设计思想更先进，更适合未来的计算发展方向，更适合我们的需求，因此我们决定选择 Flink 技术方向。

Blink – Alibaba Flink

虽然 Flink 具备流计算的各种优势，但 Flink 在成熟度和活跃度上的不足，使得我们无法在阿里巴巴业务场景中直接使用，因此我们启动了 Blink 项目，目标就是扩展、优化、完善 Flink，使其能够应用在阿里巴巴大规模实时计算场景，并将我们在阿里巴巴对 Flink 的改进都回馈给开源社区。

最近一年中 Blink 已经将多项架构、功能和性能改进贡献给 Flink 社区，例如：

Flink 架构升级，插件化原生支持不同调度系统，并实现了原生运行在 Hadoop YARN 上

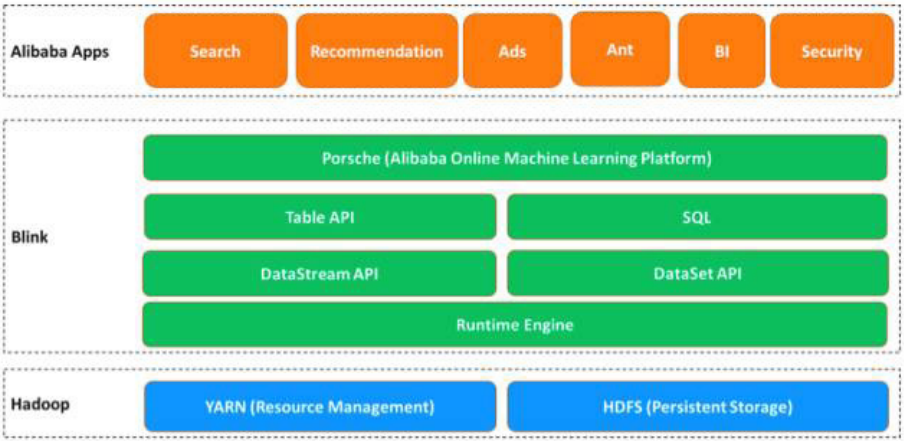
Failover 稳定性改进，优化了 Task/TaskManager 以及 JobManager 各种组件 Fail 的场景处理

提出并实现了增量式 Checkpoint 的架构，使得 Flink 的 Checkpoint/Recovery 速度大幅提升，成本明显下降

提出并实现了 Async Operator，通过异步方式，让 I/O 密集型计算节点的性能大幅提升

提出了大量 Table API 的全新设计，以及流和批在 SQL 层面的统一概念和方案

Blink 在阿里巴巴的现状



Blink 实时计算引擎在阿里巴巴内部是运行在 Hadoop 集群上的，Blink 计算任务会根据自己的需求向 YARN 申请计算资源，运行过程中周期性将计算状态持久化到 HDFS 上，以方便随时恢复，因此可以看出新型的 Blink 计算平台也可以很好的 leverage 成熟的 Hadoop 生态。

在 API 层，Blink 提供了基础的 DataStream/DataSet API，用户可以利用基础 API 有较高自由度的开发。此外，Blink 重点提供了 Table API/SQL 这种高级语言 API，可以降低门槛让更多开发人员以更低成本进行开发，这对于更多更快速的业务接入是非常有价值了，而且在 SQL 层 Flink 之前的进展是非常缓慢的，Blink 对 Flink 给与了非常及时的补充和完善。

此外，基于 Blink，我们建设出了一套在线机器学习平台 Porsche，其为算法人员提供了一套非常丰富的算法插件机制，帮助算法人员快速搭建各种常用的机器学习流程。因此，Porsche 完全 leverage 了 Blink 的实时计算能力，并释放了 Blink 在实时在线机器学习领域的力量。

目前 Blink 已经在阿里巴巴生产环境运行将近一年时间，支持了阿里巴巴多条核心业务线，例如：搜索，推荐，推荐，蚂蚁和安全等，大致的生产运行规模如下所示：

- 运行的总机器数已经超过 3000 台
- 最大的生产集群机器数已经超过 1500 台
- 每秒支持数十亿次的实时计算
- 最大的生产任务已经超过 5000 个并发，包含 10TB 级的 State，亿级 TPS

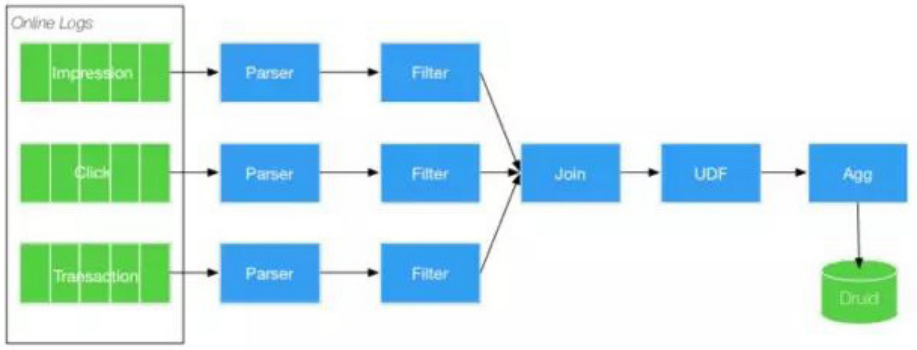
Blink 在去年阿里巴巴双 11 购物节中完成了第一次正式的挑战，搜索和推荐全实时链路全天稳定运行，不仅保证了淘宝、天猫商品实时更新无延迟，同时基于 Blink 的在线机器学习平台 Porsche 由于能够较好的将用户和商品行为实时学习，因此产生了非常好的时效性效果，大幅提升了双 11 商品成交转化率。

例如：双 11 当天有很多爆款商品，销售速度非常快，可能很快售罄，如果将用户都引导到这些商品上，会导致用户实际没有成交机会，浪费大量流量，良好的时效性数据可以让在线学习平台较快的预测到这种场景，并将用户流量进行更加合理的分配。因此可以看出，基于实时计算的在线机器学习平台其实已经开始真正走向舞台，并产生巨大价值。

Blink 在阿里巴巴的经典案例

实时 A/B Test

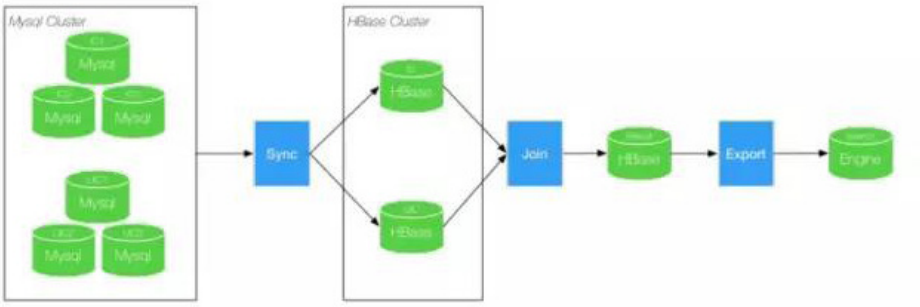
A/B Test 的目标就是通过实时数据分析和统计反馈，不断调整在线系统的算法模型，自动适应到最佳效果，A/B Test 数据收集和处理流程大致如下图所示，Blink 任务从线上日志实时同步用户行为数据，然后解析、过滤、统计，最终将各项统计指标写入 OLAP 系统中，让算法或者运营人员可以实时看到线上实际效果，从而合理调整配置各种模型，逐步达到最佳效果。



商品数索引构建流程

淘宝的搜索引擎是用户在淘宝购物的最主要入口，淘宝的商品数据处理和索引构建流程大致如下图所示，淘宝的商品库都存储的阿里巴巴的 MySQL 集群中，搜索的数据处理流程需要从 MySQL 将数据同步到搜索引擎后台的 HBase 存储中（类似：Google 都将网页抓取到 BigTable 中），然后进行各种业务逻辑处理和索引构建，最终将索引推送到在线搜索引擎中提供搜索服务。

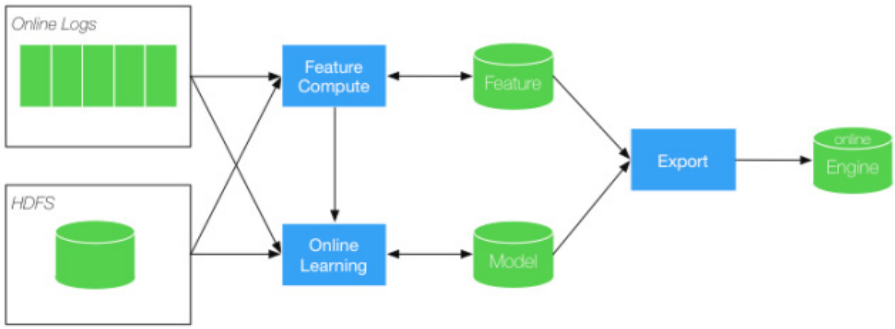
由于淘宝的商品搜索引擎需要在每天白天不断进行实时商品更新，同时晚上还需要一套额外的全量商品更新流程，因此基于 Blink 的统一计算模型，流式计算和批量计算可以使用一套用户逻辑代码完成。



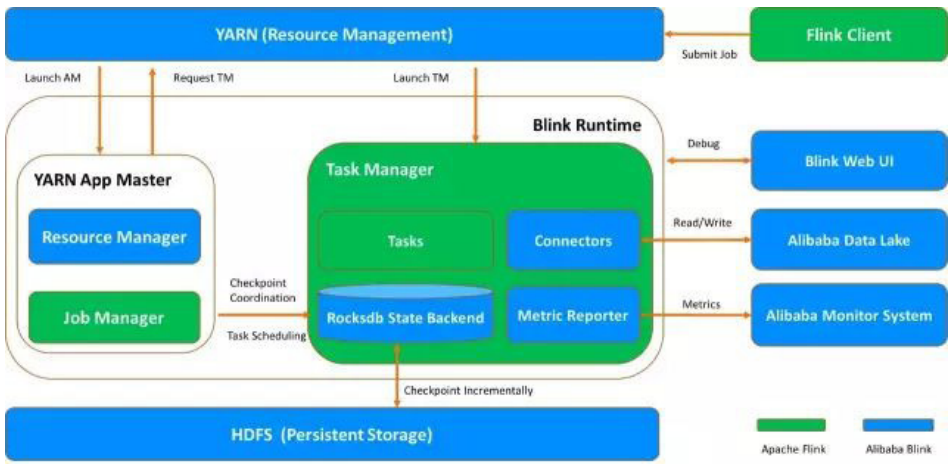
Porsche - 在线机器学习平台

在线机器学习平台利用了 Blink 强大的实时计算能力，能够实时的对海量用户和商品行为数据进行流式特征提取以及训练学习，并将实时更新的特征和模型实时同步

给在线的搜索和推荐引擎，实现个性化搜索和推荐，数据流程如下图所示：



Blink 技术架构



从 Blink 的架构图中可以看出，Blink 在内部模块组成上和 Flink 是有着非常清晰的界限的，绿色部分是和 Flink 共享的基础核心框架，Blink 在这些框架、协议和接口上的改进都会回馈给社区，保证兼容性。

但蓝色部分是扩展层，例如：资源管理，状态存储，运维监控、Debug 工具，输入输出层等，Blink 都会根据阿里巴巴技术生态和业务场景进行定制开发，使得 Blink 能够在和 Flink 共享基础内核和生态的前提下，依然能够灵活支持阿里巴巴自身的场景需求。

这种架构设计，将之前开源技术的开放通用化和企业需要定制需求的矛盾进行了

解耦，使得我们既可以和开源社区密切合作，享受开源的红利，同时也可以根据阿里巴巴自身需求进行高度定制和优化，不会受制于外部不可控因素。

Blink 的未来

目前 Blink 已经在阿里巴巴内部达成共识，成为阿里巴巴统一的实时计算引擎，接下来我们将继续加大在 Blink 技术发展上的投入，并与开源社区更加密切的合作，突进流式计算的发展。应用场景上，一方面会继续扩大计算规模，并提推出内部统一实时计算服务，统一支持阿里内部的所有实时计算业务；另一方面也将会通过阿里云的公有云和专有云渠道向外界输出我们的实时计算能力，让更多行业和用户也都能享受到阿里巴巴实时计算的技术成果。

总之，Blink 的实时计算之路刚刚开启，未来必将有更大的挑战和机遇，也非常欢迎各位对实时计算有兴趣的技术爱好者以及高校学子们投身到这件开创新一代计算平台的事情上来。

寻找热爱技术的你

阿里巴巴实时计算团队现招聘实习生，参与建设阿里巴巴新一代实时计算引擎的研发工作，技术方向涵盖分布式计算、存储和调度，工作地点北京和杭州均可，每周至少能来 2-3 天，时间持续 2-3 个月以上。要求具备扎实的计算机理论基础，较强的开发和学习能力，对技术充满热情，熟悉 Hadoop，Spark，Storm，Flink 等开源大数据技术者优先。

感兴趣的同学可以通过官网

https://campus.alibaba.com/traineePositionList.htm

申请并注明对实时计算团队感兴趣。

如何扛住 1.8 亿 / 秒的双 11 数据洪峰？阿里流计算技术全揭秘

同杰　黄晓锋

阿里妹导读：今年的双 11 再次刷新了记录——支付成功峰值达 25.6 万笔 / 秒、实时数据处理峰值 4.72 亿条 / 秒。 面对较去年增幅 100% 的数据洪峰，流计算技术可谓功不可没。今天，我们将揭开阿里流计算技术的神秘面纱。

双 11 刚刚拉下帷幕，激动的心还停留在那一刻：

当秒针刚跨过 11 号零点的一瞬间，来自线上线下的千万剁手党在第一时间涌入了这场年度大趴——从进入会场到点击详情页，再到下单付款一气呵成。

前台在大家狂欢的同时，后台数据流量也正以突破历史新高的洪峰形式急剧涌入：

支付成功峰值达 25.6 万笔 / 秒

实时数据处理峰值 4.72 亿条 / 秒

而作为实时数据处理任务中最为重要的集团数据公共层（保障着业务的实时数据、媒体大屏等核心任务），在当天的总数据处理峰值更是创历史新**高达 1.8 亿 / 秒**！想象下，1 秒钟时间内千万人涌入双 11 会场的同时，依然应对自如。

流计算的产生即来源于数据加工时效性的严苛需求：

由于数据的业务价值会随着时间的流失而迅速降低，因此在数据发生后必须尽快对其进行计算和处理，从而能够通过数据第一时间掌握业务情况。今年双 11 的流计算也面临着一场实时数据洪峰的考验。

首先来展示今年（2017 年）较去年（2016 年）数据洪峰峰值的比较：

2016 年：支付成功峰值 12 万笔 / 秒，总数据处理峰值 9300 万 / 秒

2017 年：支付成功峰值 25.6 万笔 / 秒，实时数据处理峰值 4.72 亿条 / 秒，阿里巴巴集团数据公共层总数据处理峰值 1.8 亿 / 秒

在今年双 11 流量峰值翻翻的情况下，依然稳固做到实时数据更新频率：从第 1 秒千万剁手党涌入到下单付款，到完成实时计算投放至媒体大屏全路径，秒级响应。面对越发抬升的流量面前，实时数据却越来越快、越来越准。在 hold 住数据洪峰的背后，是阿里巴巴流计算技术的全面升级。

流计算应用场景

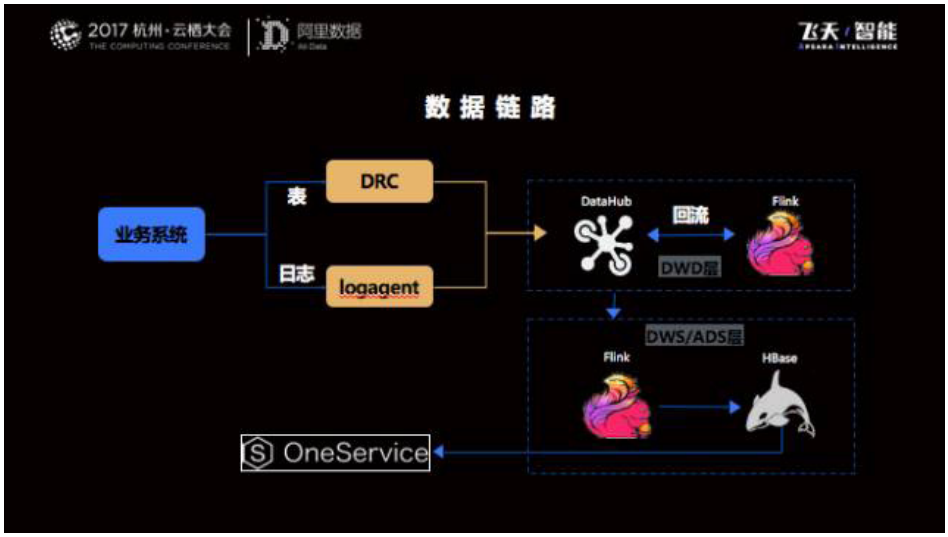
数据技术及产品部定位于阿里数据中台，除了离线数据外，其产出的实时数据也服务于集团内多个数据场景。包括今年（其实也是以往的任何一年）双 11 媒体大屏实时数据、面向商家的生意参谋实时数据，以及面向内部高管与小二的各种直播厅产品，覆盖整个阿里巴巴集团大数据事业部。

同时随着业务的不断发展壮大，到目前为止，日常实时处理峰值超 4000 万 /s，每天总处理记录数已经达到**万亿**级别，总处理数据量也达到 PB 级别。

面对海量数据的实时数据我们成功做到了数据延迟控制在秒级范围内，在计算准确率上，已实现了高精准、0 误差，达到精确处理。比如：今年的双 11 当天，双十一媒体屏第一条记录从交易表经过流计算计算处理到达媒体大屏秒级响应。

数据中台流计算实践中的数据链路

在经过最近几年大促数据洪峰的经历后，使得我们的流计算团队在引擎选择，优化性能以及开发流计算平台上都积累了丰富的经验。我们也形成了稳定高效的数据链路架构，下图是整个数据链路示意图：



业务数据的来源非常多，分别通过两个工具（DRC 与中间件的 logagent）实时获取增量数据，并且同步到 DataHub（一种 PubSub 的服务）。

实时计算引擎 Flink 作业通过订阅这些增量数据进行实时处理，并且在经过 ETL 处理后把明细层再次回流到 Datahub，所有的业务方都会去定义实时的数据进行多维度的聚合，汇总后的数据放在分布式数据库或者关系型数据库中（Hbase、Mysql），并通过公共的数据服务层产品（One Service）对外提供实时数据服务。

最近一年，我们在计算引擎和计算优化方面做了很多工作，实现了计算能力、开发效率的提升。

计算引擎升级及优化

在 2017 年，我们在实时计算架构上进行了全面的升级，从 Storm 迁移到 Blink，并且在新技术架构上进行了非常多的优化，实时峰值处理能力提高了 **2 倍**以

上，平稳的处理能力更是提高 5 倍以上：

优化状态管理

实时计算过程中会产生大量的 state，以前是存储在 HBase，现在会存储在 RocksDB 中，本地存储减少了网络开销，能够大幅提高性能，可以满足细粒度的数据统计（现在 key 的个数可以提升到亿级别了，是不是棒棒哒～）

优化 checkpoint（快照 / 检查点）和 compaction（合并）

state 会随着时间的流转，会越来越大，如果每次都做全量 checkpoint 的话，对网络和磁盘的压力非常大；所以针对数据统计的场景，通过优化 rocksdb 的配置，使用增量 checkpoint 等手段，可以大幅降低网络传输和磁盘读写。

异步 Sink

把 sink 改成异步的形式，可以最大限度提高 CPU 利用率，可以大幅提供 TPS。

抽象公共组件

除了引擎层面的优化，数据中台也针对性地基于 Blink 开发了自己的聚合组件（目前所有实时公共层线上任务都是通过该组件实现）。该组件提供了数据统计中常用的功能，把拓扑结构和业务逻辑抽象成了一个 json 文件。这样只需要在 json 文件中通过参数来控制，实现开发配置化，大幅降低了开发门槛，缩短开发周期——再来举个栗子：之前我们来做开发工作量为 10 人 / 日，现在因为组件化已让工作量降低为 0.5 人 / 日，无论对需求方还是开发方来讲都是好消息，同时归一的组件提升了作业性能。

按照上述思路及功能沉淀，最终打磨出了流计算开发平台【**赤兔**】。

该平台通过简单的“托拉拽”形式生成实时任务，不需要写一行代码，提供了常规的数据统计组件，并集成元数据管理、报表系统打通等功能。作为支撑集团实时计算业务的团队，我们在经过历年双 11 的真枪实弹后沉淀的 [赤兔平台] 中独有的功能也成为它独一无二的亮点：

一、大小维度合并

比如很多的实时统计作业同时需要做天粒度与小时粒度的计算，之前是通过两个

任务分开计算的，聚合组件会把这些任务进行合并，并且中间状态进行共用，减少网络传输 50% 以上，同时也会精简计算逻辑，节省 CPU。

二、精简存储

对于存储在 RocksDB 的 Keyvalue，我们设计了一个利用索引的 encoding 机制，有效地将 state 存储减少一半以上，这个优化能有效降低网络、cpu、磁盘的压力。

三、高性能排序

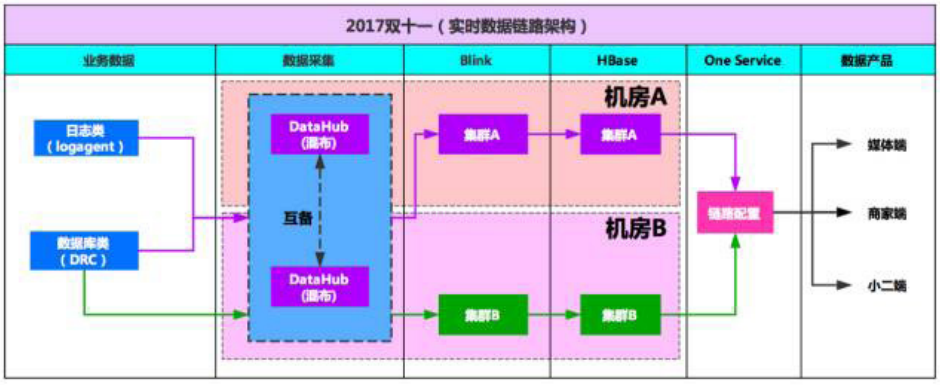
排序是实时中非常常见的一个场景，top 组件利用内存中 PriorityQueue（优先队列）和 blink 中新的 MapState feature（中间状态管理特性），大幅减少序列化次数，性能提高 10 倍左右。

四、批量传输和写操作

最终写结果表 HBase 和 Datahub 时，如果每处理一条记录都去写库的话，就会很大限制我们的吞吐。我们组件通过时间触发或者记录数触发的机制（timer 功能），实现批量传输和批量写（mini-batch sink），并且可以根据业务延时要求进行灵活配置，提高任务吞吐的同时，也降低了服务端压力。

数据保障

对于高优先级应用（每天 24 小时不间断服务），需要做到跨机房容灾，当某条链路出现问题时，能够秒级切换到其他链路，下图是整个实时公共层的链路保障架构图：



从数据采集、数据同步、数据计算、数据存储、数据服务，整条链路都是独立的。通过在 Oneservice 中的动态配置，能够实现链路切换，保障数据服务不终端。

上面内容就是保障今年双 11 流量洪峰的流计算技术秘密武器——我们不仅在于创新更希望能沉淀下来复用、优化技术到日常。

随着流计算的技术外界也在不停更迭，后续基于阿里丰富业务场景下我们还会不断优化升级流计算技术：

1. 平台化，服务化，Stream Processing as a service

2. 语义层的统一，Apache Beam，Flink 的 Table API，以及最终 Stream SQL 都是非常热的 project

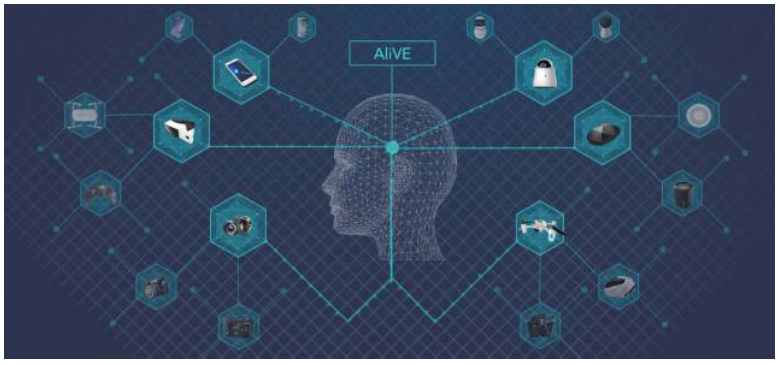
3. 实时智能，时下很火的深度学习或许未来会与流计算碰撞产生火花

4. 实时离线的统一，这个也是很大的趋势，相较于现在普遍存在的实时一套，离线一套的做法，实时离线的统一也是各大引擎努力想要达到的。

阿里知识图谱首次曝光：每天千万级拦截量，亿级别全量智能审核

览图　陈华钧　卡方

阿里妹导读：借助阿里知识图谱的建设，阿里电商平台管控从过去的“巡检”模式升级为发布端实时逐一检查。在海量的商品发布量的挑战下，最大可能地借助大数据、人工智能阻止坏人、问题商品进入阿里生态。同时面临问题商家实时的对弈、变异和恶意攻击等诸多挑战，知识图谱仍然保持着每天千万级别的拦截量，亿级别的全量智能审核次数，在滥发、侵权、合规、假货、经营范围等多个场景全面与问题卖家正面交锋，实时对弈。为了最大限度地保护知识产权，保护消费者权益，我们对知识图谱推理引擎技术提出了智能化、自学习、毫秒级响应、可解释等更高地技术要求，实现良好的社会效益。



阿里知识图谱运用

阿里巴巴生态里积累了海量的商品数据，这些宝贵的商品数据来自于淘宝、天猫、1688、AliExpress 等多个市场，同时品牌商、行业运营、治理运营、消费者、国家机构、物流商等多种角色参与其中，贡献着校正着这样一个庞大的商品库。无论是知识产权保护，还是提升消费者购物体验，实现商品数据的标准化（商品规范的统一和商品信息的确定性），以及与内外部数据之间的深度互联，意义都非常重大，阿里

商品知识图谱承载着商品标准化这一基础性，根源性的工作。 基于此，我们才能知道哪些商品是同样一件产品，我们才能确切地知道一个品牌是否被授权，品牌下的产品卖到了哪些市场。

阿里知识图谱以商品、标准产品、标准品牌、标准条码、标准分类为核心，利用实体识别、实体链指和语义分析技术，整合关联了例如舆情、百科、国家行业标准等 9 大类一级本体，包含了百亿级别的三元组，形成了巨大的知识网。

阿里知识图谱综合利用前沿的 NLP、语义推理和深度学习等技术，打造全网商品智能服务体系，服务阿里生态中的各个角色。商品知识图谱广泛地应用于搜索、前端导购、平台治理、智能问答、品牌商运营等核心、创新业务。能够帮助品牌商透视全局数据，帮助平台治理运营发现问题商品，帮助行业基于确定的信息选品，做人货场匹配提高消费者购物体验等等。为新零售、国际化提供可靠的智能引擎。

引入机器学习算法搭建推理引擎

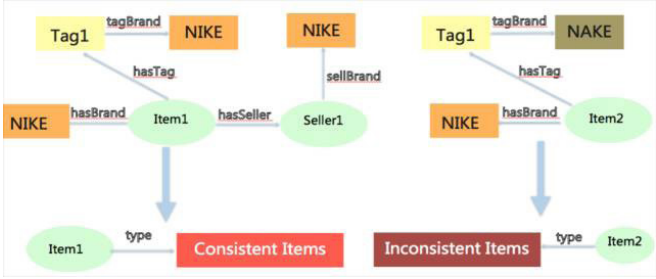
我们设计了一套框架来实现知识表示和推理。此外：知识图谱实体、关系、词林（同义词、上下位词）、垂直知识图谱（例如地理位置图谱、材质图谱）、机器学习算法模型等都纳入进来做统一的描述。

按照不同场景，我们把推理分为：上下位和等价推理；不一致性推理；知识发现推理；本体概念推理等。例如

1. 上下位和等价推理。检索父类时，通过上下位推理把子类的对象召回，同时利用等价推理（实体的同义词、变异词、同款模型等），扩大召回。 例如，为保护消费者我们需要拦截“产地为某核污染区域的食品”，推理引擎翻译为“找到产地为该区域，且属性项与“产地”同义，属性值是该区域下位实体的食品，以及与命中的食品是同款的食品”。

2. 不一致推理。在与问题卖家对弈过程中，我们需要对商品标题、属性、图片、商品资质、卖家资质中的品牌、材质、成分等基础信息，做一致性校验。比如说标题中的品牌是 Nike 而属性或者吊牌中品牌是 Nake，如下图所示，左边描述了商品标题、属性、吊牌上的品牌信息是一致的，推理为一致。右边为吊牌和商品品牌不一致

的商品，被推理引擎判断为有问题的商品。



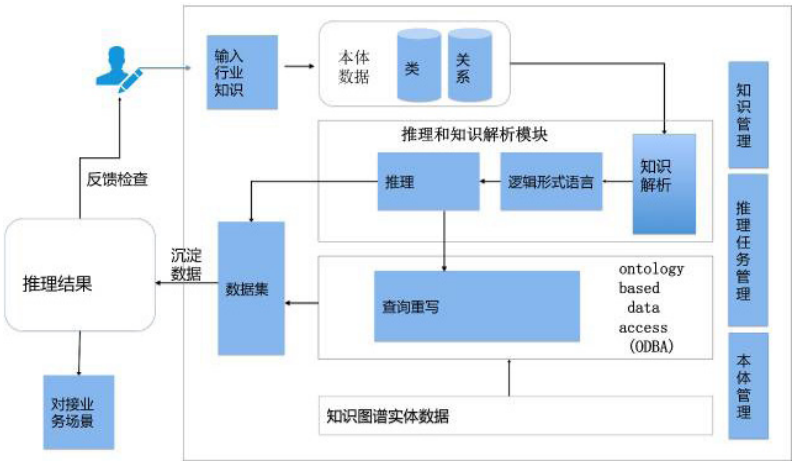
3. 知识发现推理。一致性推理的目的是确保信息的确定性，例如通过一致性推理我们能确保数据覆盖到的食品配料表正确。但消费者购物时很少看配料表那些繁杂的数字。消费者真正关心的是无糖、无盐等强感知的知识点。为了提高消费者购物体验，知识发现推理通过底层配料表数据和国家行业标准例如：

无糖：碳水化合物≤ 0.5 g /100 g（固体）或 100 mL（液体）

无盐：钠≤ 5mg /100 g 或 100 mL

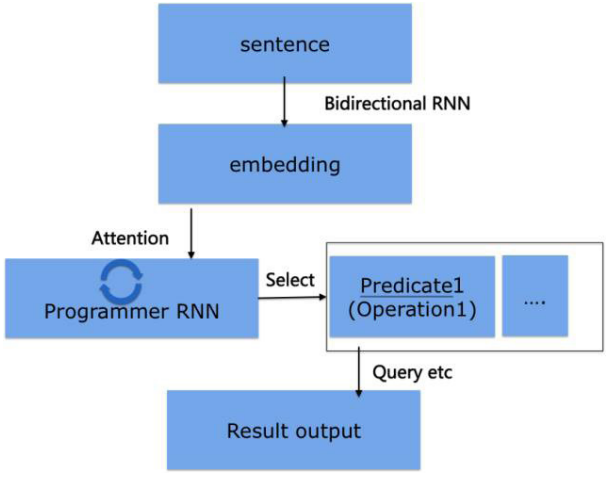
我们可以把配料表数据转化为“无糖”“无盐”等知识点。从而真正地把数据变成了知识。通过 AB test 验证，类似知识点在前端导购中极大地改善了消费者购物体验。

推理引擎背后技术框架



首先，推理引擎把自然语言通过语义解析 (semantic parsing) 转换为逻辑表达式 (logical form)。语义解析采用了结合神经网络和符号逻辑执行的方式：自然语言经过句法、语法分析、NER、Entity Linking，被编码为分布式表示 (distributed representation)，句子的分布式表示被进一步转义为逻辑表达式。

在分布式表示转换为逻辑表达式的过程中，我们首先面临表示和谓词逻辑 (predicate) 操作之间映射的问题。我们把谓词当做动作，通过训练执行 symbolico-peration，类似 neural programmer 中利用 attention 机制选择合适的操作，即选择最有可能的谓词操作，最后根据分析的句法等把谓词操作拼接为可能的逻辑表达式，再把逻辑表达式转换为查询等。过程示意如下图所示。



其次，逻辑表达式会触发后续的逻辑推理和图推理。逻辑表达式在设计过程中遵循以下几个原则：逻辑表达式接近人的自然语言，同时便于机器和人的理解。表达能力满足知识图谱数据、知识表示的要求。应该易于扩展，能够非常方便的增加新的类、实体和关系，能够支持多种逻辑语言和体系，如 Datalog、OWL 等，即这些语言及其背后的算法模块是可插拔的，通过可插拔的功能，推理引擎有能力描述不同的逻辑体系。

以上下位和等价推理为例：“产地为中国的食品”，”

用逻辑表达式描述为：

∀x: 食物 (x) ⊓ (∀ y: 同义词 (y，产地)) (x, (∀ z: 包括下位实体 (中国 , z)))

随后找同款：

∀t, x: ($ c: 属于产品 (x, c) ⊓ 属于产品 (t, c))

此外，推理引擎还用于知识库自动补全。我们基于 embedding 做知识库补全。主要思路是把知识库中的结构信息等加入 embedding，考虑了 Trans 系列的特征，还包括边、相邻点、路径、实体的文本描述 (如详情)、图片等特征，用于新关系的预测和补全。

阿里知识图谱经过我们三年的建设，已经形成了巨大的知识图谱和海量的标准数据，同时与浙江大学陈华钧教授团队成立联合项目组，引入了前沿的自然语言处理、知识表示和逻辑推理技术，在阿里巴巴新零售、国际化战略下发挥着越来越重要的作用。

有关知识图谱技术交流，或有意加入我们，欢迎联系张伟 (览图)：

lantu.zw@alibaba-inc.com

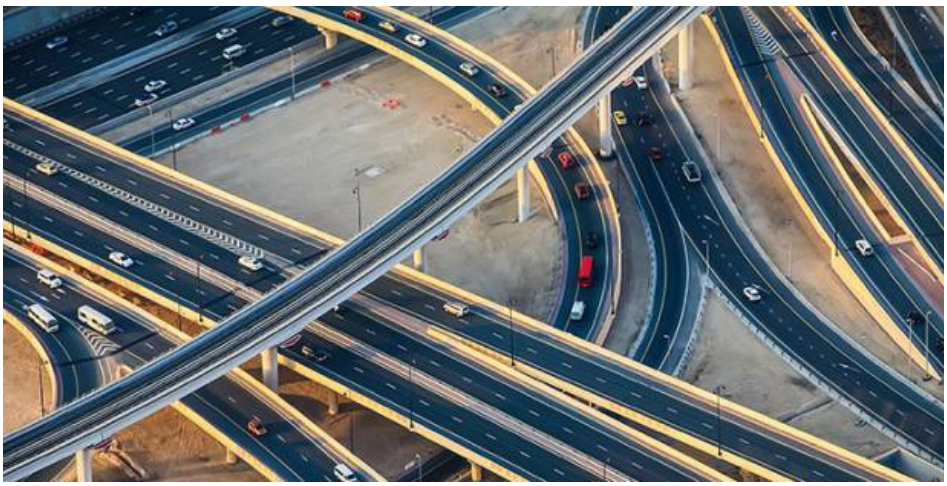
张伟 (花名：览图) 博士，阿里巴巴知识图谱团队负责人。博士毕业于新加坡国立大学，本科毕业于哈尔滨工业大学。曾任职新加坡资讯通信研究院自然语言处理应用实验室主任。

基础架构

直击阿里双 11 神秘技术：PB 级大规模文件分发系统“蜻蜓”

如柏

阿里妹导读：2017 天猫双 11，交易峰值 32.5 万 / 秒，支付峰值 25.6 万 / 秒，数据库处理峰值 4200 万次 / 秒，再次刷新了记录。阿里集团基础设施蜻蜓，在双 11 期间，对上万台服务器同时下发 5GB 的数据文件，让大规模文件分发靠蜻蜓系统完美实现。



蜻蜓，通过解决大规模文件下载以及跨网络隔离等场景下各种难题，大幅提高数据预热、大规模容器镜像分发等业务能力。月均分发次数突破 20 亿次，分发数据量 3.4PB。其中容器镜像分发比 natvie 方式提速可高达 57 倍，registry 网络出口流量降低 99.5% 以上。今天，阿里妹邀请阿里基础架构事业群高级技术专家如柏，为我们详述蜻蜓从文件分发到镜像传输的技术之路。

蜻蜓的诞生

随着阿里业务爆炸式增长，2015 年时发布系统日均的发布量突破两万，很多应用的规模开始破万，发布失败率开始增高，而根本原因就是发布过程需要大量的文件拉取，文件服务器扛不住大量的请求，当然很容易想到服务器扩容，可是扩容后又发现后端存储成为瓶颈。此外，大量来自不同 IDC 的客户端请求消耗了巨大的网络带宽，造成网络拥堵。

同时，很多业务走向国际化，大量的应用部署在海外，海外服务器下载要回源国内，浪费了大量的国际带宽，而且还很慢；如果传输大文件，网络环境差，失败的话又得重来一遍，效率极低。

于是很自然的就想到了 P2P 技术，因为 P2P 技术并不新鲜，当时也调研了很多国内外的系统，但是调研的结论是这些系统的规模和稳定性都无法达到我们的期望。所以就有了蜻蜓这个产品。

设计目标

针对这些痛点，蜻蜓在设计之初定了几个目标：

1. 解决文件源被打爆的问题，在 Host 之间组 P2P 网，缓解文件服务器压力，节约跨 IDC 之间的网络带宽资源。

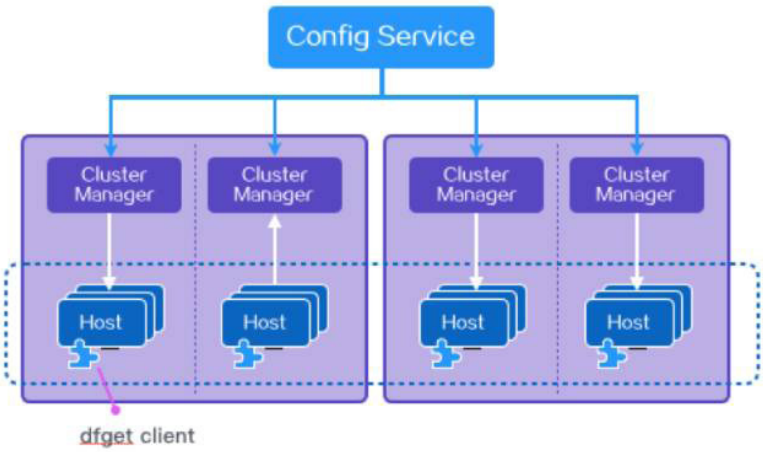
2. 加速文件分发速度，并且保证上万服务器同时下载，跟一台服务器下载没有太大的波动。

3. 解决跨国下载加速和带宽节约。

4. 解决大文件下载问题，同时必须要支持断点续传。

5. Host 上的磁盘 IO，网络 IO 必须可以被控制，以避免对业务造成影响。

系统架构



蜻蜓整体架构

蜻蜓整体架构分三层：第一层是 Config Service，他管理所有的 Cluster Manager，Cluster Manager 又管理所有的 Host，Host 就是终端，dfget 就是类似 wget 的一个客户端程序。

Config Service 主要负责 Cluster Manager 的管理、客户端节点路由、系统配置管理以及预热服务等等。简单的说，就是负责告诉 Host，离他最近的一组 Cluster Manager 的地址列表，并定期维护和更新这份列表，使 Host 总能找到离他最近的 Cluster Manager。

Cluster Manager 主要的职责有两个：

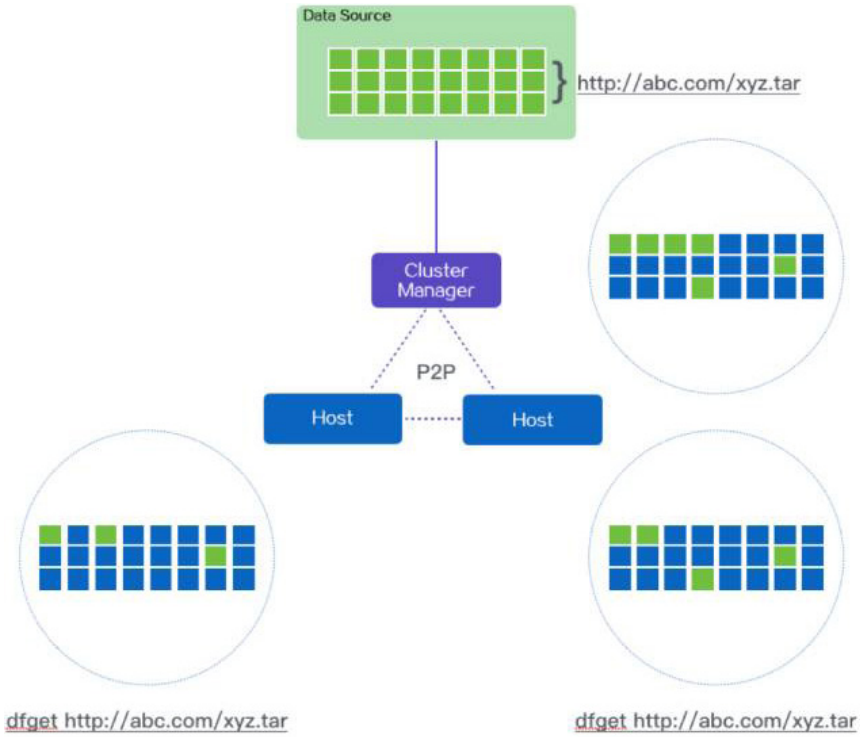
1. 以被动 CDN 方式从文件源下载文件并生成一组种子分块数据；

2. 构造 P2P 网络并调度每个 peer 之间互传指定的分块数据。

Host 上就存放着 dfget，dfget 的语法跟 wget 非常类似。主要功能包括文件下载和 P2P 共享等。

在阿里内部我们可以用 StarAgent 来下发 dfget 指令，让一组机器同时下载文件，在某种场景下一组机器可能就是阿里所有的服务器，所以使用起来非常高效。除了客户端外，蜻蜓还有 Java SDK，可以让你将文件“PUSH”到一组服务器上。

下面这个图阐述了两个终端同时调用 dfget，下载同一个文件时系统的交互示意图：



蜻蜓 P2P 组网逻辑示意图

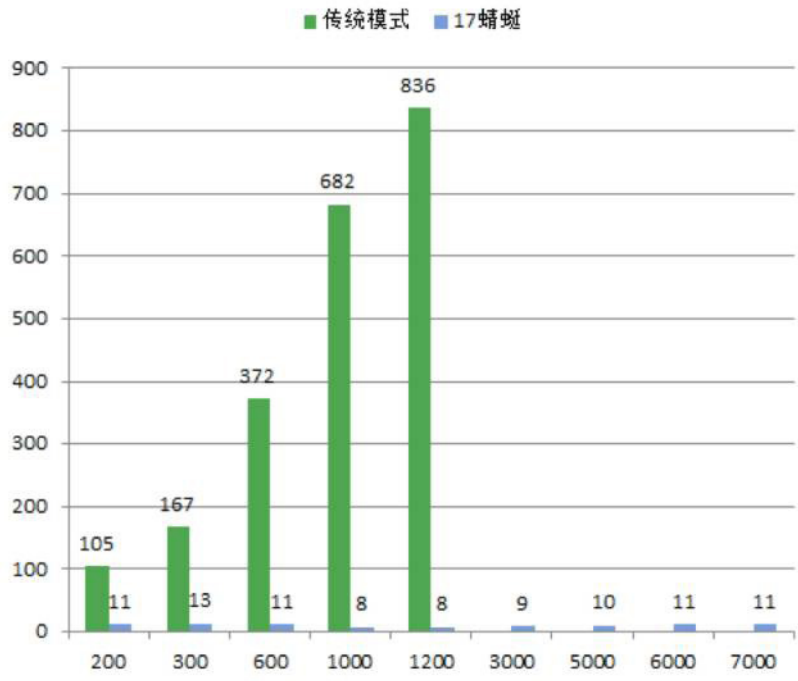
两个 Host 和 CM 会组成一个 P2P 网络，首先 CM 会查看本地是否有缓存，如果没有，就会回源下载，文件当然会被分片，CM 会多线程下载这些分片，同时会将下载的分片提供给 Host 们下载，Host 下载完一个分片后，同时会提供出来给 peer 下载，如此类推，直到所有的 Host 全部下载完。

本地下载的时候会将下载分片的情况记录在 metadata 里，如果突然中断了下载，再次执行 dfget 命令，会断点续传。

下载结束后，还会比对 MD5，以确保下载的文件和源文件是完全一致的。蜻蜓通过 HTTP cache 协议来控制 CM 端对文件的缓存时长，CM 端当然也有自己定期清理磁盘的能力，确保有足够的空间支撑长久的服务。

在阿里还有很多文件预热的场景，需要提前把文件推送到 CM 端，包括容器镜像、索引文件、业务优化的 cache 文件等等。

在第一版上线后，我们进行了一轮测试，结果如下图：



传统下载和蜻蜓 P2P 下载测试结果对比图

X 轴是客户端数量，Y 轴是下载时长，

文件源：测试目标文件 200MB（网卡：千兆 bit/s）

Host 端：百兆 bit/s 网卡

CM 端：2 台服务器（24 核 64G，网卡：千兆 bit/s）

从这个图可以看出两个问题：

1. 传统模式随着客户端的增加，下载时长跟着增加，而 dfget 可以支撑到 7000 客户端依然没变好。

2. 传统模式到了 1200 客户端以后就没有数据了，因为数据源被打爆了。

从发布系统走向基础设施

2015 年双 11 后，蜻蜓的下载次数就达到了 12 万 / 月，分发量 4TB。当时在阿里还有别的下载工具，如 wget，curl，scp，ftp 等等，也有自建的小规模文件分发系统。我们除了全面覆盖自身发布系统外，也做了小规模的推广。到 2016 年双 11 左右，我们的下载量就达到了 1.4 亿 / 月，分发量 708TB，业务增长了近千倍。

2016 年双 11 后我们提出了一个更高的目标，希望阿里大规模文件分发和大文件分发 90% 的业务由蜻蜓来承担。

我希望通过这个目标锤炼出最好的 P2P 文件分发系统。此外也可以统一集团内所有的文件分发系统。统一可以让更多的用户受益，但统一从来不是终极目标，统一的目的是：1. 减少重复建设；2. 全局优化。

只要优化蜻蜓一个系统，全集团都能受益。比如我们发现系统文件是每天全网分发的，而光这一个文件压缩的话就能给公司每天节省 9TB 网络流量。跨国带宽资源尤其宝贵。而如果大家各用各的分发系统，类似这样的全局优化就无从谈起。

所以统一势在必行！

在大量数据分析基础上，我们得出全集团文件分发的量大概是 3.5 亿次 / 周，而我们当时的占比只有 10% 不到。

经过半年努力，在 2017 年 4 月份，我们终于实现了这个目标，达到 90%+ 的业务占有率，业务量增长到 3 亿次 / 周（跟我们之前分析的数据基本吻合），分发量 977TB，这个数字比半年前一个月的量还大。

当然，不得不说这跟阿里容器化也是密不可分的，镜像分发流量大约占了一半。下面我们就来介绍下蜻蜓是如何支持镜像分发的。在说镜像分发之前先说下阿里的容器技术。

阿里的容器技术

容器技术的优点自然不需要多介绍了，全球来看，容器技术以 Docker 为主占了大部分市场，当然还有其他解决方案：比如 rkt，Mesos Uni Container，LXC 等，而阿里的容器技术命名为 Pouch。早在 2011 年，阿里就自主研发了基于 LXC 的容

器技术 T4，只是当时我们没有创造镜像这个概念，T4 还是当做虚拟机来用，当然比虚拟机要轻量的多。

2016 年阿里在 T4 基础上做了重大升级，演变为今天的 Pouch，并且已经开源。目前 Pouch 容器技术已经覆盖阿里巴巴集团几乎所有的事业部，在线业务 100% 容器化，规模高达数十万。镜像技术的价值扩大了容器技术的应用边界，而在阿里如此庞大的应用场景下，如何实现高效“镜像分发”成为一个重大命题。

回到镜像层面。宏观上，阿里巴巴有规模庞大的容器应用场景；微观上，每个应用镜像在镜像化时，质量也存在参差不齐的情况。

理论上讲用镜像或者用传统“基线”模式，在应用大小上不应该有非常大的区别。但事实上这完全取决于 Dockerfile 写的好坏，也取决于镜像分层是否合理。阿里内部其实有最佳实践，但是每个团队理解接受程度不同，肯定会有用的好坏的之分。尤其在一开始，大家打出来的镜像有 3 ~ 4GB 这都是非常常见的。

所以作为 P2P 文件分发系统，蜻蜓就有了用武之地，无论是多大的镜像，无论是分发到多少机器，即使你的镜像打的非常糟糕，我们都提供非常高效的分发，都不会成瓶颈。这样就给我们快速推广容器技术，让大家接受容器运维模式，给予了充分消化的时间。

容器镜像

在讲镜像分发之前先简单介绍下容器镜像。我们看下 Ubuntu 系统的镜像：我们可以通过命令 docker history ubuntu:14.04 查看 ubuntu:14.04，结果如下：

```
root@DaoCloud:~# docker history ubuntu:14.04
IMAGE               CREATED             CREATED BY                                      SIZE
d2a0ecffe6fa        5 weeks ago         /bin/sh -c #(nop) CMD ["/bin/bash"]             0 B
29460ac93442        5 weeks ago         /bin/sh -c sed -i 's/^#\s*\(deb.*universe\)$/   1.895 kB
b670fb0c7ecd        5 weeks ago         /bin/sh -c echo '#!/bin/sh' > /usr/sbin/polic   194.5 kB
83e4dde6b9cf        5 weeks ago         /bin/sh -c #(nop) ADD file:c8f078961a543cdefa   188.2 MB
root@DaoCloud:~#
```

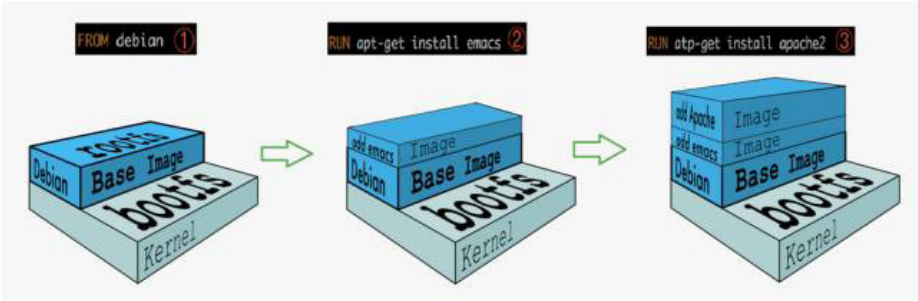
需要注意的是：镜像层 d2a0ecffe6fa 中没有任何内容，也就是所谓的空镜像。

镜像是分层的，每层都有自己的 ID 和尺寸，这里有 4 个 Layer，最终这个镜像是由这些 Layer 组成。

Docker 镜像是通过 Dockerfile 来构建，看一个简单的 Dockerfile：

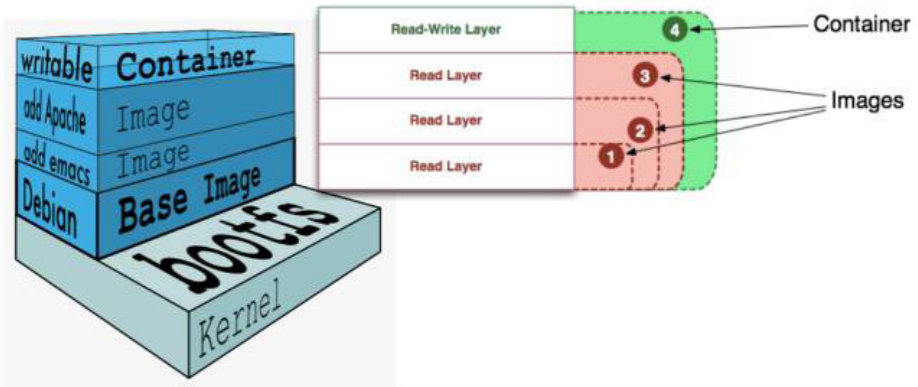
```
1. From Debian
2. RUN apt-get install emacs
3. RUN apt-get install apache2
4. CMD ["/bin/bash"]
```

镜像构建过程如下图所示：



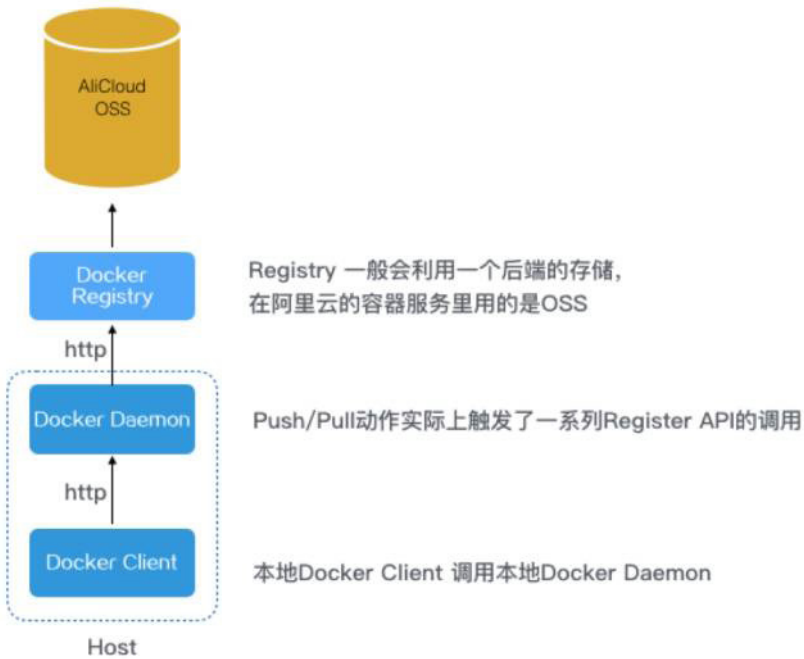
可以看到，新镜像是从 base 镜像一层一层叠加生成的。每安装一个软件，就在现有镜像的基础上增加一层。

当容器启动时，一个可写层会被加载到镜像的顶层，这个可读可写层也被称为“容器层”，容器层之下都是“镜像层”，都是只读的。



如果镜像层内容为空，相应的信息会在镜像 json 文件中描述，如果镜像层内容不为空，则会以文件的形式存储在 OSS 中。

镜像分发

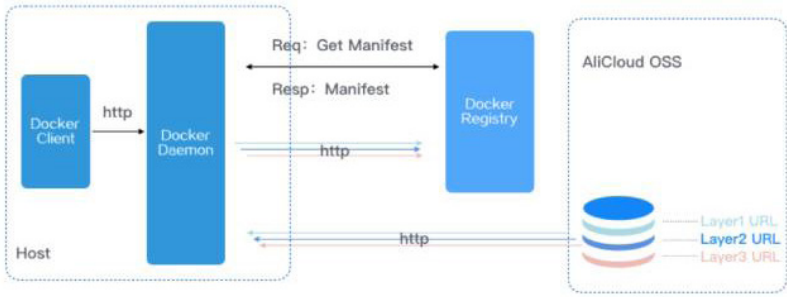


Docker 镜像下载流程图

以阿里云容器服务为例，传统的镜像传输如上图所示，当然这是最简化的一种架构模式，实际的部署情况会复杂的多，还会考虑鉴权、安全、高可用等等。

从上图可以看出，镜像传输跟文件分发有类似的问题，当有一万个 Host 同时向 Registry 请求时，Registry 就会成为瓶颈，还有海外的 Host 访问国内 Registry 时候也会存在带宽浪费、延时变长、成功率下降等问题。

下面介绍下 Docker Pull 的执行过程：



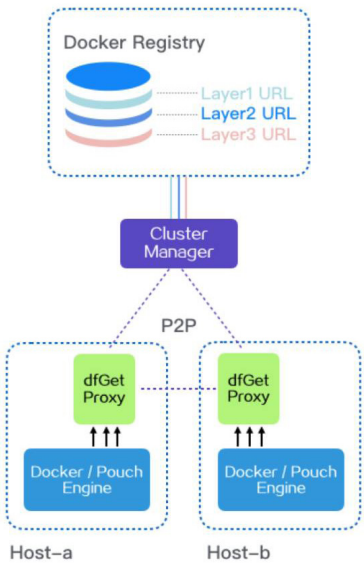
Docker 镜像分层下载图

Docker Daemon 调用 Registry API 得到镜像的 Manifest，从 Manifest 中能算出每层的 URL，Daemon 随后把所有镜像层从 Registry 并行下载到 Host 本地仓库。

所以最终，镜像传输的问题变成了各镜像层文件的并行下载的问题。而蜻蜓擅长的正是将每层镜像文件从 Registry 用 P2P 模式传输到本地仓库中。

那么具体又是如何做到的呢?

事实上我们会在 Host 上启动 dfGet proxy，Docker/Pouch Engine 的所有命令请求都会通过这个 proxy，我们看下图：



蜻蜓 P2P 容器镜像分发示意图

首先，docker pull 命令，会被 dfget proxy 截获。然后，由 dfget proxy 向 CM 发送调度请求，CM 在收到请求后会检查对应的下载文件是否已经被缓存到本地，如果没有被缓存，则会从 Registry 中下载对应的文件，并生成种子分块数据（种子分块数据一旦生成就可以立即被使用）；如果已经被缓存，则直接生成分块任务，请求者解析相应的分块任务，并从其他 peer 或者 supernode 中下载分块数据，当某个 Layer 的所有分块下载完成后，一个 Layer 也就下载完毕了，同样，当所有的 Layer 下载完成后，整个镜像也就下载完成了。

蜻蜓支持容器镜像分发，也有几个设计目标：

1. 大规模并发：必须能支持十万级规模同时 Pull 镜像。

2. 不侵入容器技术内核（Docker Daemon，Registry）：也就是说不能改动容器服务任何代码。

3. 支持 Docker，Pouch，Rocket ，Hyper 等所有容器 / 虚拟机技术。

4. 支持镜像预热：构建时就推送到蜻蜓集群 CM。

5. 支持大镜像文件：至少 30GB。

6. 安全

Native Docker V.S 蜻蜓

我们一共做了两组实验：

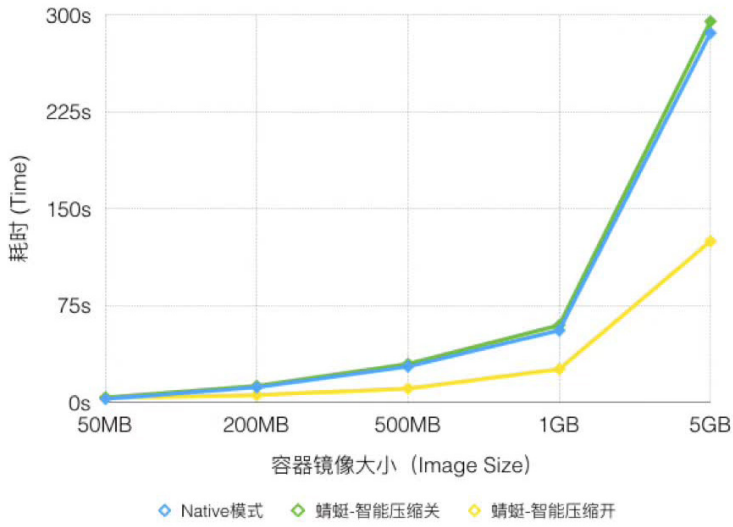
实验一：1 个客户端

1. 测试镜像大小：50MB、200MB、500MB、1GB、5GB

2. 镜像仓库带宽：15Gbps

3. 客户端带宽：双百兆 bit/s 网络环境

4. 测试规模：单次下载

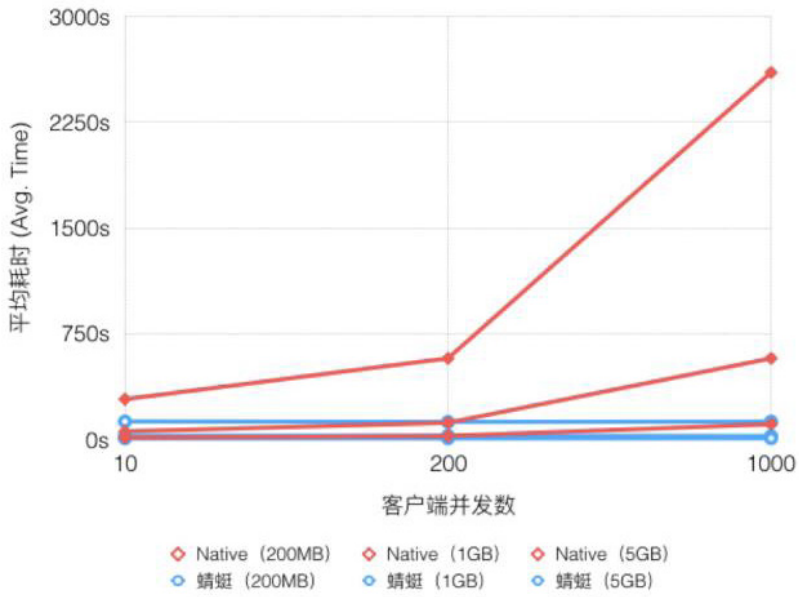


单客户端不同模式对比图

Native 和蜻蜓（关闭智能压缩特性）平均耗时基本接近，蜻蜓稍高一点，因为蜻蜓在下载过程中会校验每个分块数据的 MD5 值，同时在下载之后还会校验整个文件的 MD5，以保证下载的文件跟源文件是一致的；而开启了智能压缩的模式下，其耗时比 Native 模式还低！

实验二：多客户端并发

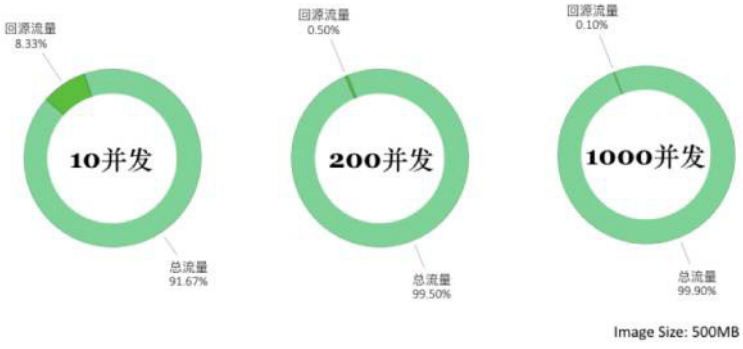
1. 测试镜像大小：50MB、200MB、500MB、1GB、5GB
2. 镜像仓库带宽：15Gbps
3. 客户端带宽：双百兆 bit/s 网络环境
4. 多并发：10 并发、200 并发、1000 并发



不同镜像大小和并发数的对比图

上图可以看出，随着下载规模的扩大，蜻蜓与 Native 模式耗时差异显著扩大，最高可提速可以达 20 倍。在测试环境中源的带宽也至关重要，如果源的带宽是 2Gbps，提速可达 57 倍。

下图是下载文件的总流量（并发数 * 文件大小）和回源流量（去 Registry 下载的流量）的一个对比：



蜻蜓镜像分发出流量对比图

向 200 个节点分发 500M 的镜像，比 docker 原生模式使用更低的网络流量，实验数据表明采用蜻蜓后，Registry 的出流量降低了 99.5% 以上；而在 1000 并发规模下，Registry 的出流量更可以降低到 99.9% 左右。

阿里巴巴实践效果

蜻蜓在阿里投入使用大概已有两年，两年来业务发展迅速，从分发的次数来统计目前一个月接近 20 亿次，分发 3.4PB 数据。其中容器镜像的分发量接近一半。

蜻蜓在阿里文件 vs 镜像分发流量趋势图

在阿里最大的一次分发应该就是今年双 11 期间，要对上万台服务器同时下发 5GB 的数据文件。

走向智能化

阿里在 AIOps 起步虽然不是最早，但是我们近年来投入巨大，并在很多产品上有所应用。蜻蜓这个产品中有以下应用：

智能流控

流控在道路交通中很常见，比如中国道路限速规定，没有中心线的公路，限速为 40 公里 / 小时；同方向只有 1 条机动车道的公路，限速为 70 公里 / 小时；快速道路

80 公里；高速公路最高限速为 120 公里 / 小时等等。这种限速对每辆车都一样，显然不够灵活，所以在道路非常空闲的情况下，道路资源其实是非常浪费的，整体效率非常低下。

红绿灯其实也是流控的手段，现在的红绿灯都是固定时间，不会根据现实的流量来做智能的判断，所以去年 10 月召开的云栖大会上，王坚博士曾感慨，世界上最遥远的距离不是从南极到北极，而是从红绿灯到交通摄像头，它们在同一根杆上，但从来没有通过数据被连接过，摄像头看到的东西永远不会变成红绿灯的行动。这既浪费了城市的数据资源，也加大了城市运营发展的成本。

蜻蜓其中一个参数就是控制磁盘和网络带宽利用率的，用户可以通过参数设定使用多少网络 IO/ 磁盘 IO。如上所述，这种方法是非常僵化的。所以目前我们智能化方面的主要思想之一是希望类似的参数不要再人为来设定，而是根据业务的情况结合系统运行的情况，智能的决定这些参数的配置。最开始可能不是最优解，但是经过一段时间运行和训练后自动达到最优化的状态，保证业务稳定运行同时又尽可能的充分利用网络和磁盘带宽，避免资源浪费。

智能调度

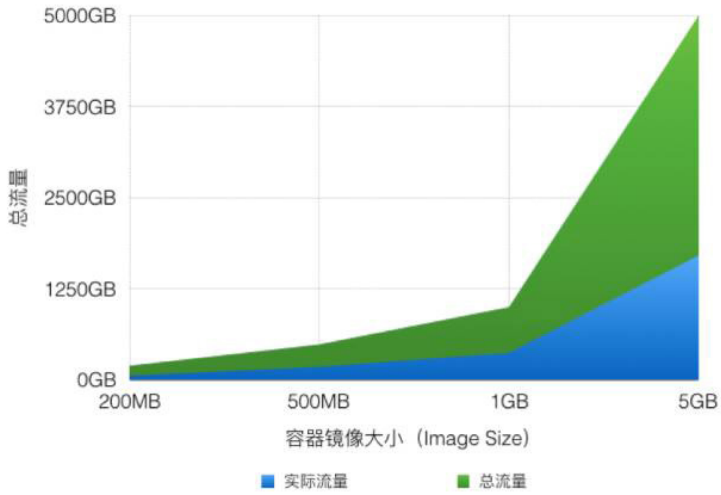
分块任务调度是决定整个文件分发效率高低与否的关键因素，如果只是通过简单的调度策略，比如随机调度或者其他固定优先级的调度，这种做法往往会引起下载速率的频繁抖动，很容易导致下载毛刺过多，同时整体下载效率也会很差。为了最优化任务调度，我们经历了无数次的尝试和探索，最终通过多维度（比如机器硬件配置、地理位置、网络环境、历史下载结果和速率等等维度的数据）的数据分析（主要利用了梯度下降算法，后续还会尝试其他算法），智能动态决定当前请求者最优的后续分块任务列表。

智能压缩

智能压缩会对文件中最值得压缩的部分实施相应的压缩策略，从而可以节约大量的网络带宽资源。

对容器镜像目前的实际平均数据来看，压缩率 (Compression Ration) 是 40%，也就是说 100MB 镜像可以压缩到 40MB。针对 1000 并发规模，通过智能压缩可以

减少 60% 的流量。



安全

在下载某些敏感的文件（比如秘钥文件或者账号数据文件等）时，传输的安全性必须要得到有效的保证，在这方面，蜻蜓主要做了两个工作：

1. 支持携带 HTTP 的 header 数据，以满足那些需要通过 header 来进行权限验证的文件源；

2. 利用对称加密算法，对文件内容进行传输加密。

开源

随着容器技术的流行，容器镜像这类大文件分发成为一个重要问题，为了更好的支持容器技术的发展，数据中心大规模文件的分发，阿里决定开源蜻蜓来更好的推进技术的发展。阿里将持续支持开源社区，并把自己经过实战检验的技术贡献给社区。敬请期待。

总结

蜻蜓通过使用 P2P 技术同时结合智能压缩、智能流控等多种创新技术，解决大规模文件下载以及跨网络隔离等场景下各种文件分发难题，大幅提高数据预热、大规

模容器镜像分发等业务能力。

蜻蜓支持多种容器技术，对容器本身无需做任何改造，镜像分发比 natvie 方式提速可高达 57 倍，Registry 网络出流量降低 99.5% 以上。承载着 PB 级的流量的蜻蜓，在阿里已然成为重要的基础设施之一，为业务的极速扩张和双 11 大促保驾护航。

PS：云效 2.0 智能运维平台 – 致力于打造具备世界级影响力的智能运维平台，诚聘资深技术 / 产品专家，工作地点：杭州、北京、美国，

Reference

[1] Docker Overview：
https://docs.docker.com/engine/docker-overview/
[2] Where are docker images stored：
http://blog.thoward37.me/articles/where-are-docker-images-stored/
[3] Image Spec：
https://github.com/moby/moby/blob/master/image/spec/v1.md
[4] Pouch 开源地址：
https://github.com/alibaba/pouch
[5] 蜻蜓开源地址：
https://github.com/alibaba/dragonfly
[6] 阿里云容器服务：
https://www.aliyun.com/product/containerservice
[7] 飞天专有云敏捷版：
https://yq.aliyun.com/articles/224507
[8] 云效智能运维平台：
https://www.aliyun.com/product/yunxiao

企业内部 IT 应用

阿里人打车不给钱？内部自研神器“欢行”首次曝光

阿里技术

阿里妹导读：业界流传着这样一则传说——阿里员工打车从来不给钱，只需手指点一点，报销流程就能自动完成。你也许忍不住会问：天底下还有这么好的事情？没错，下面就让阿里妹从产品设计、核心技术、数据等方面，为你详细介绍这款深受阿里人喜爱的出行神器——“欢行”。

欢行是阿里巴巴信息平台事业部自主研发的员工差旅 & 报销系统，涵盖用车、酒店预订、行程管理、差旅管控等功能于一体，全流程实现移动化，在提升员工出行效率的同时，大大降低了企业差旅成本，并实现了企业差旅和报销制度的系统化管控，是阿里内部最受员工欢迎的应用之一。

自动化智能化，欢行让员工出行更便利

个人免支付、免贴票、免报销

因公用车、购买机票由企业账户直接付款，员工再也不用忍受贴票、报销的痛苦，公司每月可节省近 20 万张出租车发票的审核处理成本。每天有 7000 多名阿里小二通过欢行打车，平台接入了多家供应商，能够根据当前运力和价格为员工提供最优的出行方案。

全流程移动化，随时随地欢乐出行

差旅申请、预订、报销、审批全流程实现移动化，员工不仅能随时查看公司开票资料等重要信息，还能随时查看和管理个人差旅行程。员工在外如遇紧急情况，可通过欢行直接拨打救援热线获得援助。员工在差旅期间可通过拍照、扫描的快捷方式，将火车票、电子发票等自动生成发票记录。即使是暂时无法扫描的发票，只要通过欢行预订，报销时消费记录都会自动打通，无需再次填写。


●●●●● 中国移动
下午7:42
返回
中国大陆同学国际差旅紧急救援
热线：
24小时全球旅行紧急救援热线：+86 400-888-3080
或SOS紧急救援热线+86 400-650-5718
差旅申请
移动申请，随时出行
差旅标准
酒店
协议优先，信用支付
用车
叫车搭车，一站搞定
借还款
诚信是金，信誉为上
发票
开票信息，一目了然
紧急救援
改变出差方式，让您轻松享受旅程
欢行
行程
报销
我的

差旅管控，成本管控更智能

欢行能通过低价预警系统提醒员工在合适范围内选择最低价的产品为企业节省成本，同时系统对差旅申请人和乘机人实行强关联，以防止为非阿里员工预订机票。系统还支持向不同角色（主管、财务、行政等）展示员工差旅行为和费用支出情况，并对异常情况作出提醒。

从 4 天降低到 4 小时，对账变得更简单

每个月欢行至少产生 20 万笔用车订单，针对不同供应商还有不同的结算要求，使用传统的 Excel 方式对这样数量级的订单进行对账，不仅耗时而且极易出错。而且一个财务小二在月底关帐前处理完多个业务的账务，压力极大。

欢行对账平台通过自动对接账单自动对账、自动将账单数据抛到下游财务系统进行付款、分摊、入帐等工作，财务只需核对系统对完账的数据，和供应商确认订单或金额有异常的那部分数据即可，其他一切都可交由系统自动完成，节省了大量时间（对账时间从过去的每个供应商需要花费 4 天缩减到只需 4 小时），更重要的是数据不会出错。同时系统也支持不同集团的财务处理各自集团账务的诉求。这些功能的实现，背后靠的是阿里大数据平台 ODPS。针对 ODPS 的原理，欢行对账平台的数据流图如下：

利用 ODPS 可以实现跨系统边界的数据运算。它的核心是通过 Map-reduce 进行大数据处理，提高运算速度，而传统的单库则无法支撑大数据量的运算，导致超时。另一方面，通过这种方式，新的供应商账单的接入变成了标准的流程化，对后续支持其他业务或现有业务的规则变更，有很高的灵活性，也提高了开发效率，降低了开发成本。

未来票据扫描识别，再也不用小心翼翼贴发票

报销时最繁琐的就是输入报销信息和贴发票了。虽然阿里小二们在订机票和用车时已经无需报销，但除此以外，每天仍有很多其他费用需要报销。票据扫描识别为很多小二们节省了信息录入的时间。

目前欢行已经实现或即将要实现的票据有两种：一种是火车票，通过调用图像识别服务并运用几何解析来识别车票上的有用信息，自动填入报销记录。在经过 1000+ 张的车票扫描试验后，实现了整体 83%、单项 93% 的识别率。另一种是增值税专票，同样使用了图像识别服务，加上编辑距离算法来识别发票信息。 对于较为模糊的企业名称，欢行借助搜索引擎来进行搜索和匹配企业库，使得发票的识别率能提升 10% – 30%。

打到车服务好还价格低，解决运力、服务和成本的三维平衡

欢行用车于 16 年正式上线，当时只有一家用车供应商。随着市场及内部需求的变化，公司内部对用车提出了新的需求，于是欢行接入了多家供应商，分别在运力、

服务和成本上各有所长，目标为实现运力、服务与成本的平衡。

运力：近几年，阿里在全国乃至全球的办公点越来越多，员工数量与日俱增，有一些办公点地理位置较偏僻，周围车辆不多；加上国家提高了快车的准入机制，导致供应商的运力大量下降，仅靠一家供应商的运力已无法满足小二们的出行需求。

同时，基于服务和成本的考虑，欢行转变了思路，将用车这个服务平台化，并提供标准的 API 给多家用车供应商接入。然而，与第三方系统通信，安全问题至关重要，欢行基于 TOP 平台搭建了 API 开放平台，使用签名校验的方式保障了双方通信的安全性。

服务：除了接入其他供应商来满足服务诉求，对司机服务质量的提升，欢行也做了尝试。快车，主要采用的是社会运力，司机并非自家员工，尽管供应商在司机管理上做了大量措施，仍然避免不了有司机违规作弊或服务态度恶劣。

欢行使用 ODPS 进行大数据分析捕获并在系统层面建立了一系列规则，只要司机命中其中一条或几条规则，欢行会自动将司机加入黑名单，阻止黑名单司机再次接到阿里小二们的单。同时，为了防止司机骚扰小二们，欢行协同供应商，采用传递小号的方式，司机看不到小二真实的号码，但可通过 APP 和小二取得联系。

成本：去年，市场上对企业端实行了溢价策略，在车辆不足的情况下，系统通过

自动给司机加价来吸引司机接单。阿里小二加班用车需求比较集中，预估价接口调用过频，会让服务商系统以为该区域需求旺盛，从而触发更高的溢价倍数。为了抑制这种情况，欢行利用滑动窗口算法对供应商预估价接口调用频率进行动态调节，溢出的需求会提醒小二们稍后再试。通过这种方式，溢价成本下降了 20%。

欢行的初衷是为员工提供更简单、便捷的差旅 & 报销体验，未来我们将继续致力于通过智能化出行 & 报销、OCR、电子发票等手段持续提升员工的差旅 & 报销效率和体验。我们期望更多的开发者参与建设欢行差旅 & 报销平台服务更多的企业。

同时，欢行的开发者平台即将到来，欢迎大家收藏网址：https://open-hatrip.alibaba.com，关注我们的动态。

淘宝天猫背后，有一个你不知道的神秘组织

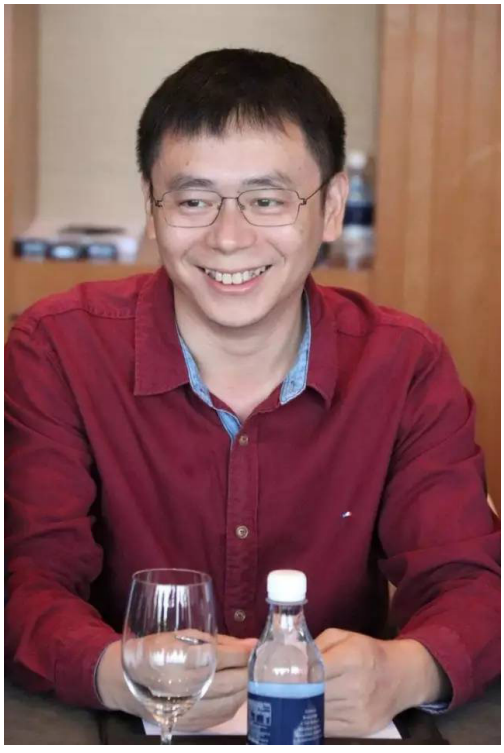
阿里技术

对大部分人来说，生活中最熟悉的是那些贴近日常的服务：不论是淘宝、天猫，还是优酷、闲鱼，它们和我们最密切，因为你可能天天都会用到。但当提到阿里巴巴电商业务中台时，我相信身边大部分人会感觉很陌生，的确，这不是一个我们会接触到的部分。

所以我们不妨换个说法，当你可以在闲鱼上能直接登录淘宝、支付宝账号，或者发现在刷微博时能直接跳转到淘宝和天猫店铺时，是不是感觉这些跨平台服务的对接都非常顺畅，阿里巴巴生态服务扩张的同时，这些不同类型的服务为什么能够如此密切的联系在一起？这恐怕是很多人好奇的地方。

这就得提到「中台」了，围绕「买买买」这件事，阿里巴巴不断诞生了新的服务，如果说这些服务像人的脸面一样特色鲜明，让所有人都能找到最属于自己的那一款的话，那中台就像是经络一样串联起这些服务，并为它们提供连接阿里不同模块之间的能力，从这个意义上来说，淘宝、天猫为你带来了便利，而「中台」则给淘宝、天猫带来了便利。

这样一个部门对于很多人来说却是神秘又陌生的，因为它对接的并非用户，而是阿里大大小小的生态服务。从 2015 年 7 月马云提出「大中台」战略以来，阿里用了近两年时间建立阿里巴巴大中台，形成「大中台，小前台」的组织形式，这对阿里扩展边界起到了举足轻重的作用，这背后，阿里经历了怎样的演变?「大中台，小前台」战略到现在效果究竟如何？神秘的「中台」对我们的生活产生了哪些影响？近期，阿里巴巴业务平台负责人玄难，在媒体采访中回答了这些问题。



阿里巴巴业务平台负责人玄难

新经济时代下的「发动机」

「大中台，小前台」的核心战略最早提出是在 2015 年 7 月。彼时阿里巴巴有近 4 亿用户，服务超过 1000 万各类企业，业务种类繁多，业务之间相互网状依赖。团队众多，相互依赖，对业务响应也越来越慢，这是内部原因。

而外部原因，则是大数据和云计算逐渐成为新经济时代的「石油」与「引擎」时，阿里巴巴更迫切的需要找到能够对外界变化快速反应，整合阿里各种基础能力，高效支撑业务创新的机制。这样，「大中台」的建设迫在眉睫。

在外界看来，「大中台，小前台」最直接的表现是打破了阿里过去的组织边界，将庞大而复杂的技术和组织关系梳理清楚，形成了「大中台」，更好的支撑「业务小前台」快速决策、敏捷行动。提出了大中台思路之后，阿里对组织也做了一些调整，而在玄难看来，最核心的其实是思想变化：

提出中台战略之后，阿里巴巴集团 CTO 行癫的要求是聚集力量把关键技术做深做厚，避免低水平重复建设，又要有良性的竞争。在 IDC、数据平台、计算平台、语音处理、图像处理、搜索、业务平台这些核心能力建设上都有更长远的战略布局和资源投入。这样对整个公司的基础能力建设有了比较大的提升，对前端的业务的响应速度各方面快了很多。

Q：业务中台的前身是共享平台，这种转变背后的原因是什么？

玄难：名称变化的后面还是思想变化。原来共享平台更多的被当做资源团队，承接各个业务方的需求，为业务方在基础服务上做定制开发。而业务中台的目标是把核心服务链路（会员、商品、交易、营销、店铺、资金结算等等）整体当做一个平台产品来做，为前端业务提供的是业务解决方案，而不是彼此独立的系统。

因为阿里巴巴的生态非常复杂，很多业务方本身也很年轻，要怎么去做，下层到底能提供什么样的支撑是不清楚的。当有大中台思路之后，第一，我们这个体系里有什么样的能力，可以让各业务很清楚的知道，也可以让前台业务方更快的理解、选择和使用中台能力。第二、我们提供了基础解决方案，业务方根据需要做定制开发满足自己的业务特性，对前台的业务来说会更快。就像我们最近上线的一个新业务 2 周完成，而按原来的模式可能要 2 个月。

当然大中台的建设思路，对系统架构提出了更高的要求。必须对系统进行重新设计，实现「业务逻辑和平台逻辑分离，不同业务之间的逻辑隔离」，才能让前台业务方可以自主定制开发。当然团队定位不一样，结果不一样，团队的成长也完全不一样。所以说共享跟做中台有很本质的区别。

Q：大中台战略之后，这种转变的提升效果怎样？

玄难：作为大中台的团队，我们提供基础服务和解决方案。具体的定制需求我们会尽量让业务方自主完成，减少大量团队沟通，提升效率。也避免原来共享团队成为资源瓶颈的问题。我们的职责是不断的提升基础平台能力基线，让所有业务方都有更好的效率。如果业务方确实需要，我们也会冲上去和他们一起快速搞定业务。

就像原来所有交易需求都需要我们团队一块去参与建设，现在 90% 以上的交易相关需求都由业务方在业务中台上自主开发，自主上线。我们能看到所有业务逻辑，发现共性的东西，就抽离成平台基础能力，让所有业务方都可以获得。

这样业务平台团队的工作节奏也逐渐可以自主把控，有更好的业务思考和规划，为前台业务提供更多更好的基础能力。

新能力的「连接者」

在阿里提出「大中台，小前端」战略之后的这近两年时间里，阿里巴巴生态的「基础设施」构建进一步完善。一个明显变化是，在生态边界扩张的同时，阿里的基础服务组件，交易、营销、会员、电子凭证、资金、店铺、评价等商业运作的环节都被体系化打磨好，作为完整的解决方案被用于各个创新业务当中。而相对应的，新业务能够在直接利用这些解决方案的同时提出新要求，并对这些服务进行针对性创新，从而进一步促进平台创新。

这种相辅相成的动态体系，成为了中台业务的「分享经济」。因此中台不再是单纯的技术、资源提供者，反而通过其他业务的创新能力「反哺」，形成了新的，更有价值的基础能力，在阿里集团内，大中台成为了「共创共建，互利互助」的平台。

Q：大中台战略实施之后，阿里在一些业务上的反应更迅速了，怎样理解这种迅速？

玄难：互联网领域的变化是非常快的，而且是一个复杂的生态，所以依赖的东西也很多。大中台这个思路理顺之后，让业务方，原来可能需要三个月才能上线的新业务，现在变成一个月、几周，上线的周期极大的缩减，这本身是非常直接的体现。

除了效率以外，因为阿里生态的业务非常多，单独一个业务能做的事情相对就

有限。但有了中台后，其实我们会把所有的业务能力，都通过底层进行连接。比如说原来电商跟优酷可能是独立的公司，后来做了整合，但这两个业务看上去是比较独立的，我们通过中台的权益中心，把优酷会员变成电商的一种营销资源，提供给淘宝天猫的商家去赠送给消费者。

比如说我买一个电视，赠送一个优酷的会员「包年」，买了一本书，可能也赠送一个优酷的会员「包月」。这种连接是中台来推动，把很多业务打通，让业务和业务连接。有时候一个独立领域的创新，它要做突破很难，但是跨领域的连接产生的创新，反而是更多的。在阿里这个生态里，通过中台能够让不同的业务之间产生连接，这其实对业务的创新是巨大的支撑。

Q：业务中台和数据中台是什么关系，是平行的还是包含的?

玄难：中台是一个基础的理念和架构，我们要把所有的基础服务用中台的思路建设，进行联通，共同支持上端的业务。业务中台更多的是支持在线业务，数据中台提供了基础数据处理能力和很多的数据产品给所有业务方去用。业务中台、数据中台、算法中台等等一起提供对上层业务的支撑。

Q：也就是说，不论是业务中台还是数据中台，实际上都是一个架构层面的去连接底下这部分资源。

玄难：对，就是能力，基础的能力，能力把它很好的呈现给我们业务方，而且对业务方业务的实施我们有一些建议，有一些基础的方案供他们选择。

新生态的「加速器」

Q：您之前提过阿里在电商系统发展上有四个阶段，第三个阶段是「三淘」，第四个阶段就是变成了现在的「业务中台化」，中台战略它最终要解决哪些问题，它会是一个最后的阶段吗?

玄难：我们讲整个淘宝最早是一个系统搞定，后来不行，必须分拆，用分布式架构，后来每个系统又很复杂了。阿里的生态太大了之后，其实每个人进来已经不知道阿里有什么了，所以必须通过中台把我们有什么能力要呈现出来，让业务方根据自己的需要去选择去使用。同时，我们在架构上能让业务方在这些能力上可以自己去定制，组装成自己的业务。当前的问题通过中台的思路去解决，慢慢这个矛盾就会变低，但必然会产生新的矛盾，就需要用新的思想去解决。

总之，任何一个企业或者一个体系，在不同的阶段遇到的主要矛盾是不一样的。但随着时间的发展，原来的主要矛盾就变成次要矛盾，产生新的主要矛盾，需要用新的方案来解决。一句话就是说，中台一定不是最后一个阶段，只是解决当下的核心矛盾的一个思路。

Q：大中台的建设，和阿里巴巴的「五新」战略有什么样的关系？

玄难：「五新」是马老师提出的战略方向。而这些战略需要我们去落地实施，需要更强大的技术体系去支撑，而「大中台、小前台」是承接「五新」战略最好的技术战略和组织形式。

Q：业务中台现在一共有多少人？他们原来和现在做的事情有什么不一样的地方？

玄难：业务中台总共 400 多人，承载了阿里基础业务服务，包括：会员、商品、交易、营销、店铺、详情、资金、结算、财务等等，每年双 11 最核心的交易链路都在我们这个团队支撑。

原来他们按系统独立去对接业务。但是按中台思路我们是一个整体，我们要把所有系统打通链接好，变成一个产品，一种能力去对外提供服务。现在我们提供统一的业务服务界面，业务方要解决什么业务问题，我们可以给提供整体的解决方案，而不是之前业务方来找我们要资源，要单独开发个什么功能，要几个人支撑一下等等。

Q：业务中台会进行新业务的孵化吗？怎样做？

玄难：我们会做一些新业务的孵化。因为我们能看到所有业务状况，也知道业务的困境在哪里。

比如，我们在数据上，孵化「地动仪」，就是新零售中基于 LBS 的客户数据化运营解决方案，这些东西都是我们自己驱动的。其实，某种意义上讲业务中台也是集团的「战略机动部队」。在公司需要的时候，会组建新业务支撑团队。以前的 O2O、百川，现在新零售的盒马鲜生，天猫海外业务等等最早都是由业务平台承建孵化，在基本成型后再把它独立出去发展，同时我们也会给新业务输出人才。

Q：在未来，业务中台在阿里生态中还会有什么样的作用，会怎样发展？

玄难：业务中台，有四件事情肯定要去做：

第一，保证阿里的业务跑得更快，更稳定。比如保障双十一稳定，同时不断提升前台业务的开发效率。

第二，产生创新。有三种形式，一种是我们看到了某个业务模式比较好，我们会把它变成一种基础能力，提供给更多业务方用；第二种是打通业务之间的连接，例如把阿里生态中 A 业务和 B 业务连接，提供给客户新的价值；第三种是通过自己的思考形成新的产品能力，就像前面提到的「地动仪」。

第三，根据集团的需要进行新业务孵化。孵化到一定阶段，觉得可以独立发展的，我们分拆出去，就像我们现在重点投入的海外市场。

第四，人才的培养。在中台，我们能看到集团所有的业务，同时也支撑所有的前台业务，相对来说系统性思考，全局性思考会更好一些。所以，我们也会根据需要给前台业务培养和输出人才。

阿里怎么发工资？自研薪酬管理系统首次曝光

墨逐

人力资源管理系统是用集中的数据将几乎所有的人力资源相关的信息（组织、招聘、薪资、绩效、审批等）统一管理起来，是企业运行必不可少的管理软件。国际上知名的有 Oracle PeopleSoft、SAP 和 Workday HCM，世界 500 强公司有超过一半都在使用。国内金蝶和用友在 eHR 领域也有着多年的技术积累。

阿里早在 06 年就上线 Oracle PeopleSoft HCM 系统，是国内最早一批引进世界先进人力资源管理软件的企业之一。PeopleSoft HCM 的上线为阿里带来了先进的人力和组织管理能力，支撑阿里业务和人员规模在过去这十年中成倍扩张。

当前，阿里经济体已经成长为涵盖电商、云计算、文娱、体育等数十家公司，数万员工覆盖 24 个国家和地区，上线十年的 PeopleSoft HCM 在功能模块、用户体验、系统开放集成等方面已经不能满足业务发展要求。再者，从自身信息安全和成本角度来考虑，阿里巴巴这种体量的公司除了自行研发系统之外，也别无选择。

PeopleSoft 界面

独立自研面临四大挑战

自研系统听起来是美好的，但要真正实施起来面临的挑战是难以想象。

挑战一：去 PeopleSoft HCM 涉及的模块众多，关系错综复杂，依赖紧密，人员信息管理，绩效系统，调薪系统、薪资申报平台，报表中心，权限系统等等，10 年历史数据都需要清理和迁移而且不能有一点点差错，否则损失不可挽回。

挑战二：国内外根本无可参考借鉴的去 PeopleSoft 公司，规模大的公司不用 Oracle 就切换到 SAP 或 Workday，小公司业务深度浅去除后换同类软件就可以了。阿里不但体量大，而且作为互联网科技公司业务场景比传统大企业还要更深，仅功能模块需求就有 70 多个。

挑战三：阿里经济体在迅速壮大，如优酷、UC 等企业不断加入，国际化进程也在加速，自研系统就必须要支持生态化，国际化需求，牵扯业务方更是繁杂。

挑战四：自研系统在成本上还要可控。HCM 管理软件是技术和业务高度融合的复杂业务系统，很多世界一流科技公司有技术实力进行研发但缺少业务深度的重要一环，以及后期维护和技术更新带来的变动，投入大量人力物力与换来的价值不匹配。阿里作为一家企业，也需要解决这个问题。

分阶段开发借助成熟技术

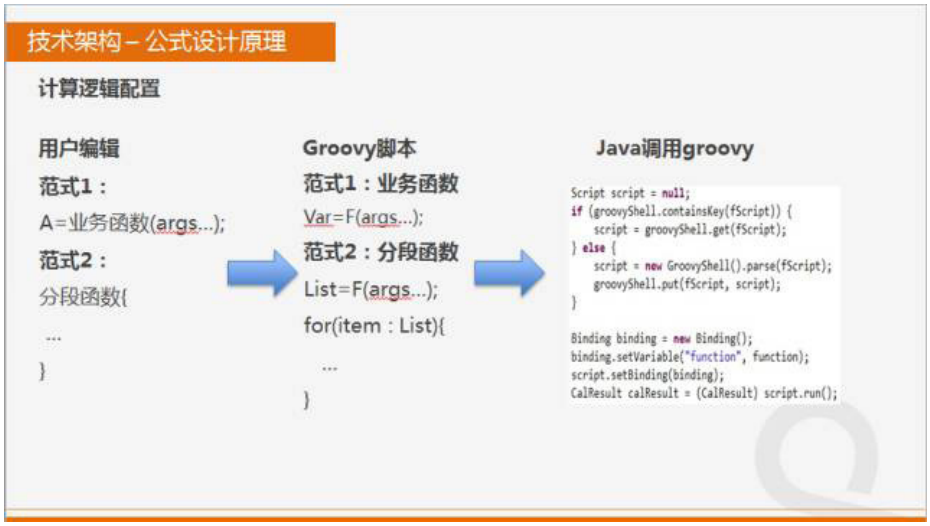
面对挑战，阿里信息平台 eHR 技术团队采取分阶段，由易到难，逐步迁移的策略。在正式启动去除前，将公司组织、职务体系、员工生命周期等主数据管理率先从 PeopleSoft HCM 中剥离出来，与集团 IT，财务、行政、采购、安全、廉政等系统打通，实现核心数据备份存储。

项目启动后，分四个阶段由浅入深进行，完成方案设计论证、计算框架开发、核算逻辑开发、并行验证等任务。

方案设计论证阶段。集合了阿里众多资深的薪酬业务和产品专家，开始去 PeopleSoft 薪酬核算的攻坚之路，在 2 个多月的时间内先后完成了脚本语言（Python/Groovy）选型，追溯方案设计论证，计算框架设计，云计算任务调度设计，数据加密等。

计算框架开发阶段。为了实现复杂业务易于管理和维护并和计算框架分离，业务代码实现上我们选择使用 Groovy 脚本语言配合阿里云大数据计算服务数加（MaxCompute）实现薪酬核算。其中对于脚本语言的选择，主要基于两方面的考虑，其一 Groovy 于 Java 无缝兼容；其二 Java 工程师快速上手 Groovy 开发，相对成熟，学习成本低。而利用阿里云大数据计算服务数加（MaxCompute）实现薪酬核算，可以经济高效的分析海量数据，用于阿里数万员工的薪酬核算在数据安全和计算效率上相较于之前都会有很大的升级。

为了让业务方和开发能快速理解整个系统的业务逻辑，以及对代码版本进行控制和管理，我们在项目公式的设计上分为三层结构，第一层是业务能看懂的业务语言，第二层是系统语言，第三层是可执行的 Groovy 脚本语言；当用户在页面编辑保存第一层的业务语言时，相对应的会转换成系统语言和 Groovy 脚本进行保存，计算时只有 Groovy 脚本参与计算。



技术架构 – 设计原理

核算逻辑开发阶段。完成了实习生薪资（100+ 计算项目），股权计税（100+ 计算项目），正式员工薪资（200+ 计算项目）计算三个迭代发布，突破了阿里员工休假晚提补报、出差、月中入离异、欠款、无息贷款利息计税、福利补贴，社保公积金基数变更，股权等十多项复杂薪资业务的计算效率与准确性瓶颈。

并行验证阶段。2016 年 9 月自研系统正式进行为期 8 个月的双系统并行验证，在这过程中同时完成了 6 个月追溯期数据重建以及最重要的年终奖计算发放的双线验证。

阿里信息平台 eHR 技术团队自主重构了 HR 领域内 PeopleSoft HCM 使用到的所有功能模块，还开发了包含简历优选，阿里学习，股权管理系统等 40 多个产品的更全面的功能模块，其中薪酬核算仅用个位数的研发人员在短时间内完成了开发到发布，真正实现技术和效益双突破。

今年 5 月，阿里关闭所有 PeopleSoft HCM 系统的同步接口，自研薪酬系统正式为阿里员工提供服务，系统支持移动办公并实现了 24 个国家（地区）和几十个家阿里经济体公司的统一管理。

效率提升 6 倍，成本降低百倍

借助阿里云的大数据计算服务数加（Maxcompute），阿里自研系统在国内第一个做到基于云端的薪资计算（基于阿里 Maxcompute），全集团数万人只需 30 分钟，PeopleSoft 同等资源下需要 3.5 小时，计算效率提升超过 6 倍并且随着员工数和数据量的增加，计算时间并不会有太大波动，也解决了将来的扩容问题。此外，通过自研系统的上线运行，每年仅授权费用就节省数百万，其他相应的维护管理费用也出现成倍的降低，总体成本实现了百倍降低。

计算过程可视化，系统简单可维护

自研系统在薪资项目、适用群组、计薪周期、计算规则与公式、发放审批流程、计算结果报表输出全链路，生态公司接入并行管理等十几个应用场景，真正体现了互联网产品的简洁易用，清晰明了，稳定高效的产品理念。

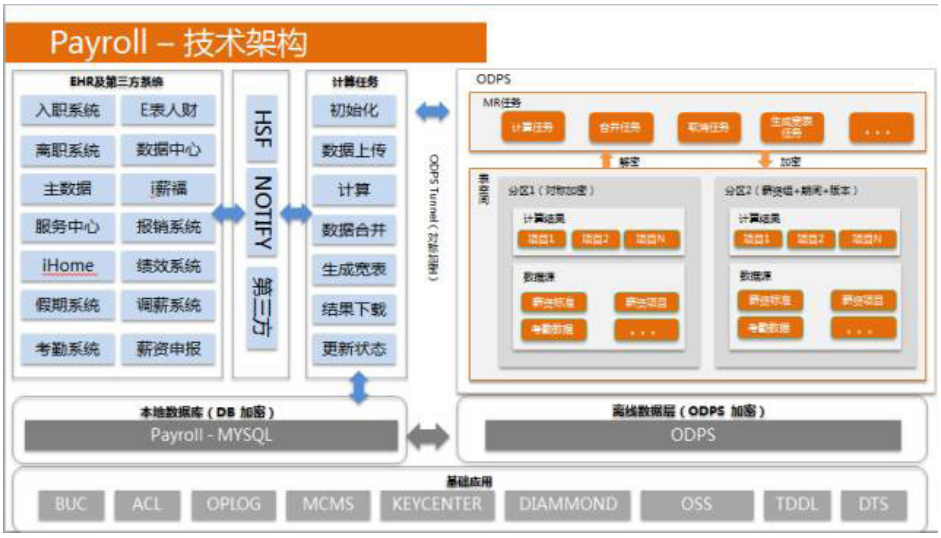
系统除了支持追溯和分段计算的功能，还支持在整个计算链路中间过程数据以及异常差异数据都可直观实时透视，相对于 PeopleSoft 及业内其它 eHR 产品，无技术背景的业务方也能快速了解整个系统的计算逻辑以及快速定位解决问题，维护简单。

追溯计算过程可视化

平台化架构，功能和场景更加丰富

自研系统对业务逻辑进行了高度抽象，通过页面配置 Groovy 脚本，在计算时将数据源和计算逻辑都同步到 Maxcompute 完成计算。薪酬计算框架好比是自动化流水线，只要提供了数据源和计算逻辑就可实现想要的结果，这种计算框架和业务逻辑的分离使得它能够实现的功能和场景更加丰富。

目前该系统除了已经支持薪酬计算，13 薪，在国内首创基于薪酬核算框架做到同时支持员工股权归属和行权的税务自动计算等多个业务模式。



自研系统平台化架构

数据加密，更关注员工隐私

自研系统本地数据库 Mysql 使用阿里加密服务 KeyCenter 进行敏感字段加密存储，开发和数据库管理员都无法接触到数据，避免原来 PeopleSoft 系统数据库明文存储问题，更加安全可靠同时也关注保护员工的个人信息。

员工体验获得极大提升

自研系统支持手机端和 PC 端，集成员工基础档案、薪资档案、每月工资，奖金、股权，社保公积金，年度收入等，实现员工随时随地查看薪资发放结果、个税、

福利补贴，销售佣金、社保基数调整等等信息，还支持一键求助、快速 12 万报税，员工体验获得极大提升。

自研系统信息查询界面

未来智能 HCM 系统

实现 PeopleSoft HCM 系统的下线是新技术场景驱动智能办公的重要一步，阿里 eHR 技术团队将会在全球化、生态化、移动化、数据化、服务化 5 大方向做深做透，基于阿里系大数据、算法、云计算平台打造出服务于阿里全球经济体的智能 eHR 系统。

如何像阿里工程师一样高效办公?

祁越

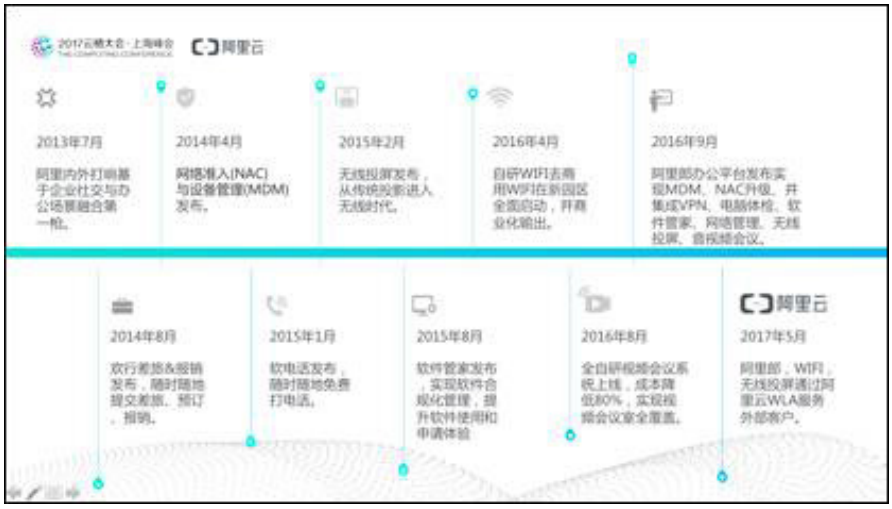
上周，阿里巴巴总市值达到 3565 亿美元，再次超越腾讯，成为亚洲市值最高的公司。在阿里庞大的经济体量背后，究竟是何种智能、高效的信息化系统支撑着它不断前进？阿里人究竟是如何办公的?

在 6 月 10 日召开的云栖大会 · 上海峰会上，阿里巴巴信息平台总监祁越，为大家揭开了阿里的智能化办公之路。

以下是演讲全文：

大家好，我叫祁越，是阿里巴巴信息平台 –IT 负责人。今天很荣幸跟大家分享阿里的智能化办公场景。

阿里巴巴信息化实践



阿里的办公信息化始于 2013 年，阿里巴巴信息平台事业部推出了“阿里内外”——以企业主站的形式为员工提供最基本的搜人、寻找站点入口等服务，这是信息化的第一步。

紧接着随着公司快速成长，我们意识到，如果所处的网络不安全、没有数据沉淀，那一切所做的信息化工作都是徒劳无功的。所以在 2014 年 4 月，信息平台启动了 NAC（网络准入）和 MDM（设备管理）并进行了落地。

之后，我们又遇到了新的挑战。当时阿里入职的每位新员工都会配备 1 台 1000 多元的固定电话，而这个固定电话 90% 的时间是不用的，所以我们启动了“软电话”项目，员工只要拿着电脑或者手机，就可以随时随地免费打电话了。

2015 年随着越来越多的技术同学加入阿里，公司给员工配备了 Mac 笔记本，但是没有投屏的接口，信息平台研发了“无线投屏”，无论 Mac ／ Windows 笔记本还是手机，都可以随时随地把桌面投到电视上。

2015 年之前，很多员工会上网下载免费软件，下载后却发现软件里自带一堆的流氓软件，且无法卸载。所以我们做了“软件管家”，员工可以在上面获取工作所需的各种软件。通过软件管家把所有的办公软件实行规范化管理，软件申请实现自动化

审批，并可以随时随地在后台看到每天的软件使用量、申请量、剩余库存等等，实时进行管理。

2014 ~ 2015 年期间，阿里园区的发展速度非常快，仅杭州地区一年就有六七个园区项目在进行，且每个园区都是千人以上的规模。面临的问题就是当时阿里用的是商用 Wi-Fi，每一个新园区都需要布置控制器，再配一个网工，因为除了网工，没有人能让 Wi-Fi 工作，基本一个园区要花费几个人做一个月，而上线以后网工还要做各种用户支持。

于是我们的研发团队用了 1 年多时间，做自研 Wi-Fi。一开始心里是打鼓的，因为要把业界非常成熟的 Wi-Fi 产品替代成自己 5-10 人团队开发出来的产品，研发同学们心里还是没底的。然而就在当时一个非常关键的时刻，园区的某个商用 Wi-Fi 的控制器突然挂了（那个关键时刻可能就是双十一接近零点的时候），当时让所有人下定决心，立刻启用了自研 Wi-Fi。

近两年来，随着阿里的园区、办公点越来越多（包括阿里国际化以后，美国、英国、东南亚、印度等地区），对音视频会议的要求非常高。一开始我们用的是业界 TOP3 的硬件品牌，但它操作起来比较复杂，遥控器上至少有 30 多个按键，导致员工开视频会议前，需要先找 IT 预约指导。为了优化体验，让员工能够随时随地拿着电脑、手机开会，我们又开启了全自研的音视频会议之路。

现在，信息平台把之前做的所有基础办公产品进行整合，建立了统一的办公平台——内部叫“阿里郎”。到今年，一个特别重要的里程碑就是，刚才提到的所有产品，都将通过阿里云来服务于我们的外部客户。

企业信息化建设

我们的很多产品都围绕着以下几点进行：国际化、生态化（让外部公司、投并购公司也能使用）、稳定性、智慧化（用户只要点一下就能解决各类复杂问题）、移动化（无论是开会还是投屏，都能移动办公）。

阿里的智能化办公基本围绕着以下三方面去做：

第一，体验。买一个商用产品交给用户，这不是体验；只有当产品能让用户产生惊喜，这才是体验。

第二，成本。产品在成本方面是不是很有竞争力，是不是能让公司的 CFO 对成本控制、对所研发的产品能为公司省的钱而点赞。

第三安全。

云 Wi-Fi

Wi-Fi 是一个硬件，阿里作为一家互联网公司为什么要做硬件？前面提到的阿里在企业信息化之路上遇到的困难和挑战：无论是成本、稳定性还是运维的工作量，都需要一个智能化产品来解决这些问题。但环顾业界并没有合适的商用产品，需要静下心来自己研发。

信息平台自研的云 Wi-Fi 有哪些价值呢?

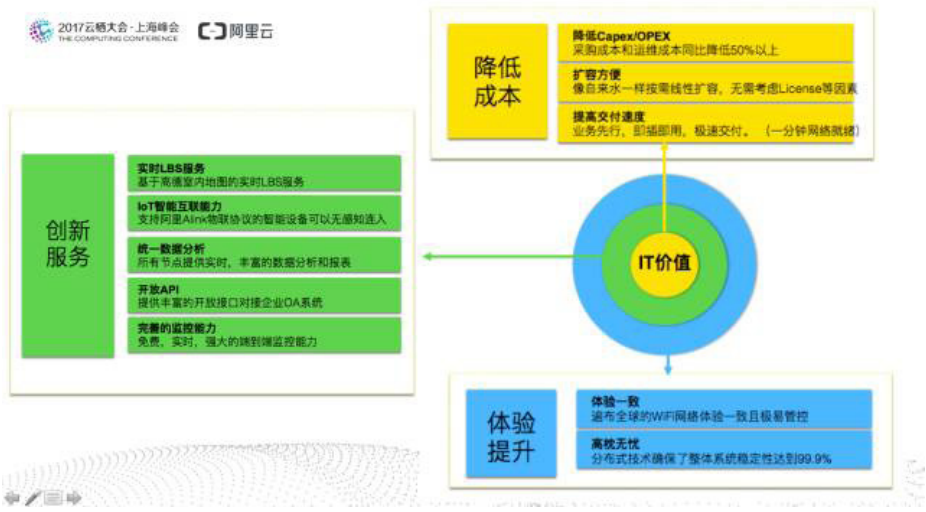
第一，成本。相比原来的商用产品，云 Wi-Fi 能节约 50% 以上成本。

第二，便捷。不依赖于网工，只要拿着一个 AP，找一个网口插上，员工的网就通了。员工从 A 办公室漫游到 B 办公室再到 C 办公室，全程无缝体验。

第三，可用性。所有传统 Wi-Fi 的架构都有个致命的问题，就是有控制器。只要有控制器就会在整个网络链路上增加故障点，在关键时刻会成为致命的瓶颈。

第四，灵活。在互联网时代，有非常多的 APP 需要跟 Wi-Fi 进行连接。譬如某企业来了访客，装了某个 APP 就可以自动连上 Wi-Fi。这些在商用场景下很难办到，但今天阿里的 Wi-Fi 可以轻松实现。

第五，智能。今天基于云 Wi-Fi，可以作为很多物联网玩法的一个天然网关。如果想分析今天员工的开会情况，在工位的情况，上下班考勤的情况，每个会议室的利用率情况等等，用云 Wi-Fi，结合后台的数据分析就可以实现。



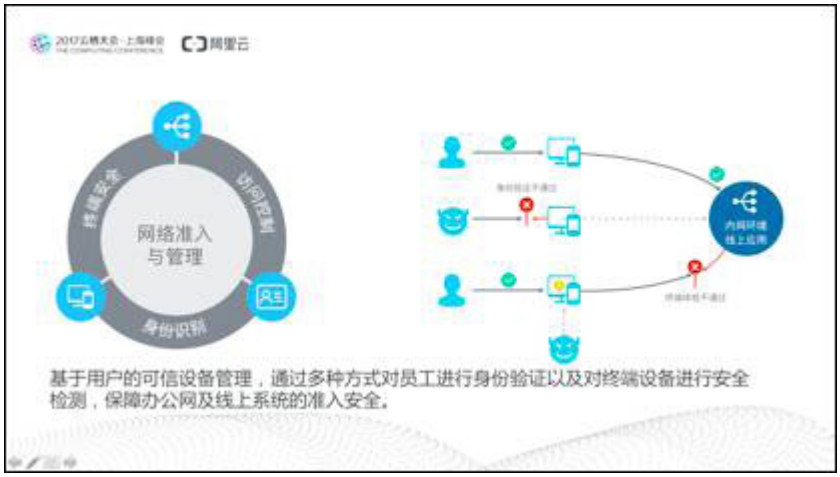
Wi-Fi 可以很好的体现 IT 的价值。IT 的价值是什么?

第一，降低成本。除了给用户带来惊喜的体验，一定要用更少的人，更低的成本，为企业创造更高的价值。

第二，体验提升。当员工、老板，走到公司里的任何一个 office，看到同样的 ID、热点而不需要输入任何密码的时候，这种体验是非常好的。在从办公室到茶水间到会议室甚至到食堂的过程中，可以始终拿着手机办公。

第三，创新。在互联网时代，需要硬件设备来支持很多的玩法和打法，Wi-Fi 可以作为一个很好的切入点帮助我们实现 LBS 室内地图、数据分析、大数据采集和分析等，这些玩法需要大家一起去思考。

网络准入和设备管理



其实这两个词听起来还是比较生硬的。为什么要做设备管理？因为在阿里非常在意对网络的轻管控。如果把所有的权限从用户侧收走，把所有的数据锁在保险箱里，网络一定是安全的，但没有任何意义和生产力可言。

设备管理，即知道接入的设备是谁，它的主人是谁，以及你做了什么。如果做了不该做的事情，就会有响应机制来解决。今天我们通过“MDM 和“NAC”来实现对设备的识别、对人的识别、对动作的识别。环顾业界的商用产品，远远满足不了员工的需求。

譬如说，很多产品会在一个设备有风险的时候，一脚把它踢掉，但是在阿里不行，因为有很多同学会做测试，做实验，如果你把它踢掉，业务会受影响。只有把它拉到一个我们认为安全的“VLAN”里去，既不影响公司的其它业务系统，也能继续访问互联网，只有非常开放的系统才能做到。

网络准入还有一个点就是 VPN。VPN 是离开办公室之后一个很重要的办公场景。左边是传统的相对安全的 VPN 方式：令牌加上登录的账户，但有个非常严重的问题，令牌加密码加用户名，基本上没有人能一次输入正确，百分之五六十的用户需要输两次才能成功。在阿里，当你想登陆公司内网的时候，只要在电脑和手机上分别点一下，一共点两次，就连接上 VPN 了。我们统计了下，这样的接入方式，每个月为阿里节省了五六千个小时。

云投屏

这是阿里园区一个很普通的会议室，只有一个电脑、电视、以及一个视频会议摄像头。今天我们已经把无线投屏做成了阿里标配的办公方式，无论员工想把屏幕投到当前会议室的电视机上，还是让异地的同事看到屏幕，都能轻松实现。如果员工要开会，没带电脑，只带了手机，而手机里有要投的 PPT，依然可以在手机上点一下自动投屏。

有员工觉得投屏码比较麻烦，于是我们增加了声波投屏的方式，只要带着电脑进入会议室，电脑就能自动识别投屏码，员工只需点一下投屏按钮就能把屏幕投到最近的电视上。

我们把办公场景下很多需要 30s 完成的步骤缩减到只需要 2s，这就是给用户更好的体验。目前统计了一下，无线投屏每个月可以为阿里巴巴节省至少六千个小时。

音视频会议

今天，信息平台支持着阿里的全球化办公，各地办公区的很多诉求是外部的商用产品满足不了的。以遥控器为例，如果开音视频会议需要配备一个上面有 30 多个按键的遥控器，那估计有百分之五十的用户不会使用它（包括我自己）。但今天在阿里巴巴的任何一个音视频会议室，是见不到遥控器的。

说到成本，音视频会议的成本一直很高，我们的产品做到了将成本控制在业界 TOP3 产品的一半左右，而且比业界的产品更灵活，比如能够支持 1000 人的会议、电话入会、无缝对接传统的视频会议设备等等。

我们还可以看到音视频会议对降低差旅的贡献：随着海外视频会议的使用量持续上升，与之对应的是集团海外差旅在同步下降，这也是这个产品的业务价值。

这就是之前提到的“阿里郎”，把内部的多个产品整合到了这里。PC 端有网络管理、电话、会议、投屏、电脑管家等，移动端有软令牌，以及其它例如会议日程、对智能设备的感知等 IOT 的应用。

当然，刚刚介绍到的信息平台旗下产品都会通过阿里云来输出给外部客户和企业，让所有企业员工能够和阿里巴巴员工在一个平台上办公。

阿里云办公架构

以上是阿里云办公的整体架构图，从统一的企业控制台、终端 APP，再到支持各种能力的智能化办公产品（包括沟通、协同、基础设施等），以及底层的 IAAS 和 PAAS 服务等，可以让企业能以非常低的成本使用和阿里一样的办公方式。

未来已来，谢谢大家！

没想到，阿里工程师每天必刷的网站是……

阿里技术

阿里妹导读：在阿里内部，有个不为外人所知的协作平台——阿里内外。经过四年发展，许多创新的想法、产品从阿里内外走出，而阿里内外也从 0 做到如今近百万 PV。究竟阿里内外是如何带来组织生命力？背后又有哪些核心技术？

阿里人每日必逛的神奇内网

阿里内外是阿里内部员工使用的企业运行与协作平台。它诞生于 2013 年，彼时只是一个门户和企业社交的入口。但经过 3 年发展，阿里内外实现了平台化运营，不仅接入众多阿里应用与系统，阿里的生态公司也开始享受阿里内外提供的一体化服务。今年，阿里内外开始向 3.0 智能模式发展，通过互联网数据和算法技术，增加诸如企业搜索、企业推荐、智能工作辅助，通过智能模式提高员工协同办公效率。

阿里内外界面

阿里有一句老话：一个人可以走得很快，但是一群人可以走得很远。在阿里，组织文化与工作协同是最重要的两大核心生态，作为服务内部员工的协作平台，文化和协同也是阿里内外不可或缺的核心元素。

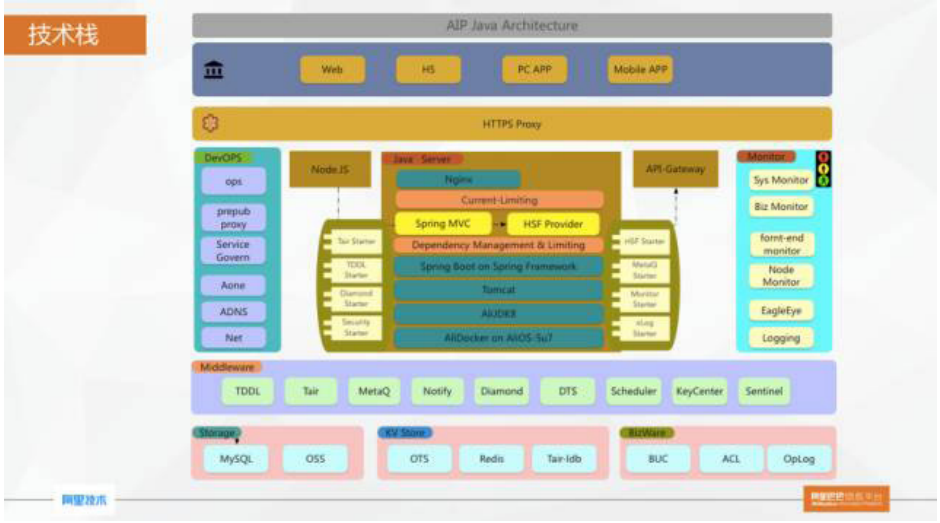
在组织文化方面，阿里内外上有一个非常具有阿里特色的版块——阿里味。阿里高管和员工都愿意在阿里味上分享自己的点子和想法，甚至是组织上的一些问题也可以畅所欲言，大大激活了员工的想象力。此外，通过阿里学习、内外直播等版块，一些技术大牛和产品大牛也会经常把好的经验分享给内部员工，帮助大家一起更好成长。

当然，在交流之后，员工最终还是需要聚焦于自己的工作本身。在工作协同方面，阿里内外还为员工提供了众多办公协同产品，如答疑、任务跟踪、周报笔记、文档、团队协作等。员工可以通过一站式搜索快速定位产品，将所有工作内容形成沉淀，大大提升工作效率。最关键的是，所有数据沉淀后，员工在一年内的工作成果会自然而然地在平台上有所体现，赋予组织更多生命力。

那么，在技术上，阿里内外是如何实现组织文化与工作协同服务的？下面将通过阿里内外技术栈、搜索架构、Feed 流、以及全球部署架构四个方面进行解读。

站在巨人肩上阿里内外技术栈一览

在技术栈方面，阿里内外站在巨人的肩膀上，复用阿里巴巴集团的技术栈体系，并基于如上的方案进行创新、新技术快速试验来提升研发效率，如 Spring-Boot、Spring-MVC、Hystrix。



简单来说，整个阿里内外技术栈可以用“三横两纵”来描述。最上面的一“横”是统一接入层。主要提供统一 Https 管理、Https 加解密以及 Https 的卸载。通过它到达下层后，都变成了 Http 协议。

第二个“横”是应用层。应用层中很多内容，均基于阿里技术实现，如 AliOS、AliJVM、AliTomcat。在此基础上，也引用了 Spring- Boot 作为整个开发框架，同时 Spring-Boot 中还放入很多插件，如 Tair 插件，数据库插件等，总数多达二十余种。底层则经过 Spring- Boot 封装，把日常的中间件、数据库、权限认证都放入这一体系中，这样开发人员在进行业务操作时，能更多聚焦于业务开发上。

最下面的一“横”是要去复用集团中间件和云上的中间件，如大家熟悉的 TDDL、Tair、MetaQ、OTS、Redis 等等。同时在这一层还有信息平台内部的业务中间件，如帐号、权限体系、操作日志等。

“两纵”分别在应用的左右端。左端是 DevOPS 的运维体系，同样采用阿里集团整套运维体系；右端则是监控部分，包括有系统监控、应用监控、前端监控，以及 node 监控，也会运用到 EagleEye 作为全链路监控体系，和日志采集记录系统。

阿里内外搜索框架

正如之前的介绍，在阿里内外上内容源多且类型复杂，不仅有文档信息，还有丰富的组织信息、应用系统。如何根据用户搜索进行快速意图识别，成为阿里内外搜索最大的技术挑战。

阿里内外的技术人员将整个搜索架构分为三层来做。最底端是数据层，用于内容源对接。目前阿里内外主要提供三种对接方式：离线对接 ODPS，诸如人事制度等不经常更改的内容，通过 ODPS 进行对接；数据接口，由搜索提供 API 接口，内容源来进行定制推送；数据库 dump，由搜索直接去对应业务的数据库，来做数据增量 Dump 等。

中间一层为搜索引擎，这是基于阿里巴巴自研 ISEARCH 做员工、文档、站点的搜索引擎；同时运用图数据库 NEO4J 来实现组织架构树，以及员工亲密度、关系网的建设。最上层则是 Query 意图识别与一些应用场景。

由于阿里内外丰富的内容源和内容类型，Query 排序十分复杂。为实现更好得搜索排序，阿里内外的 Query Planner 复用了许多阿里技术的中间件。基础服务有 PAI 算法计算平台、ODPS 离线数据计算分析，公共组件用到很多算法的组件，如分词等，具体的功能和应用场景有搜索词分类、拼写纠错、下拉提示等。

由于在阿里内部，所有用户的登陆都是实名的，因此有很多数据可以做分析，这样就可以达到“千人千面”的效果，比如技术人员搜索关键字后，系统会对其更多展示技术相关内容。

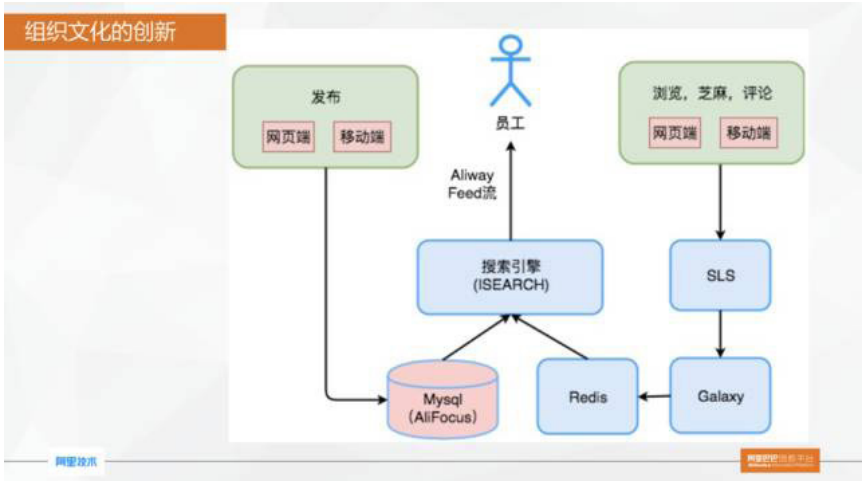
阿里内外搜索还有个功能就是“即搜即用”，即在不用二次跳转的前提下，可以

快速呈现微应用进行使用。这是因为阿里内外开放了一个应用中心让集团各产品系统进行接入。这样当用户搜索对应关键词时，就可直接进入该应用中去。

引入热度 Feed 流创新组织文化

一般公司在做组织文化管理时通常用到 BBS 论坛的方式进行。其中最大弊端就是帖子根据最后更新时间排序，这会导致很多信息混杂在一起，不利于信息聚焦。

为解决这一问题，阿里内外引入具有热度的 Feed 流。

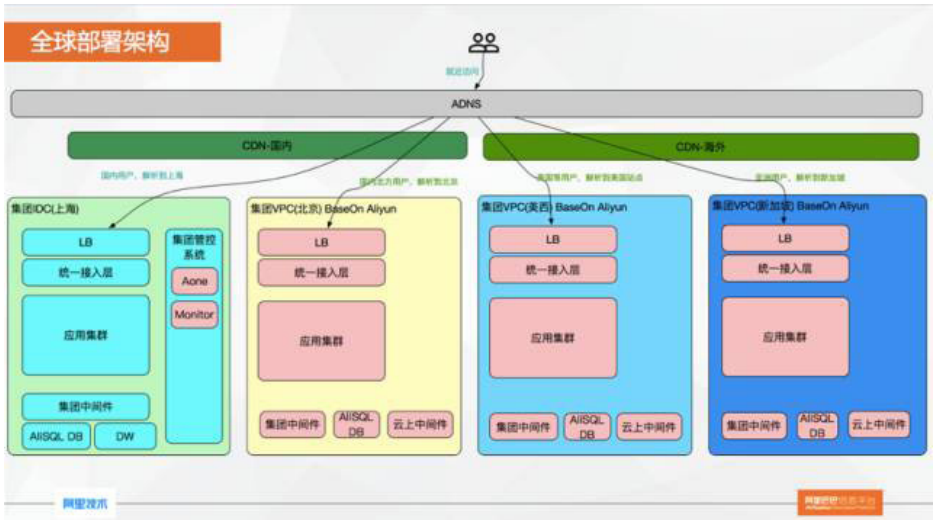


正如之前介绍，“阿里味”是员工在阿里内外上进行文化交流的主要阵地。而“阿里味”则主要由三部分组成：帖子，帖子的发起人，帖子的分类。发帖人通过 Web 端或者移动端发帖子时，中间会有一个算法的文本分类，将帖子分类到相应的板块，并把数据存放在数据库中，用户浏览时操作会记录到日志里面，日志的动作又会流入到计算平台，计算平台会针对数据进行热度分析，分析后的数据存入 Redis 中进行热度排名，会根据用户的浏览习惯呈现出不同的排序结果。

全球部署海外员工的“丝般顺滑”体验

阿里巴巴在全球分布员工近 6 万人；在美国、英国、澳大利亚、法国、德国、印度、俄罗斯、新加坡、阿联酋、甚至一些中东国家都设有工作室。在这种情况下，全

球访问和就近访问成为阿里内外为海外员工服务的挑战。



对此，阿里内外运用阿里全球 ADNS 能力，以及阿里云分布在全球各地 IaaS 基础设施，让系统可以在全球进行站点接入和部署。为了数据一致性，我们通过利用阿里云数据传输服务 DTS 来做数据的备份。这样即便身处国外的同学在访问阿里内外时也能收获“丝般顺滑”的上网体验。

目前，出于安全考虑，阿里内外只对阿里巴巴内部员工进行开放，但经过阿里内外团队对产品的不断优化，在未来，这一阿里员工才能访问的神奇内网，也会通过阿里云或钉钉等渠道，逐步开发部分产品功能给所有人，让大家也能一起体验阿里内外的独特魅力。

阿里技术

扫一扫二维码图案，关注我吧

「阿里技术」微信公众号

「阿里技术」官方微博