众微科技 WeCREDO

众微致成（北京）信息服务有限公司
WeCREDO (Beijing) Information Service Co.' Ltd

多模态数据在金融业的发展进展与落地

信用金融服务体系运营商

见微知著

大纲

CONTENTS

大纲

CONTENTS

大纲

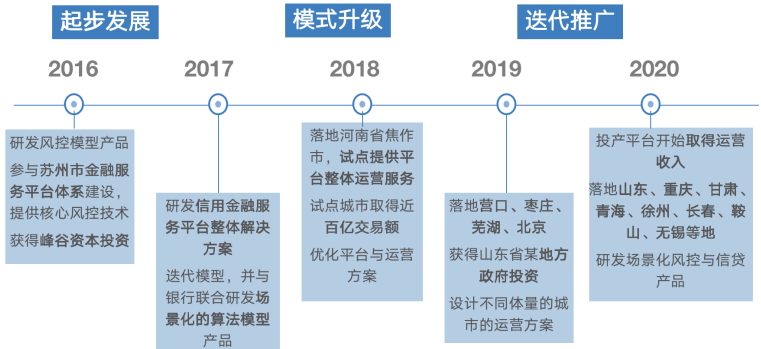
公司简介

众微致成（北京）信息服务有限公司（以下简称“众微”）成立于 2015 年 1 月，专注于提供面向信用与金融服务的大数据、人工智能等相关技术、产品与平台的研发与运营服务。核心成员均来自于国际评级公司标普、京东数科、IBM 等知名公司。公司及团队服务过的机构包括中国银行、中国建设银行、中信银行、平安银行、浦发银行、华为、百度、三星、亚洲开发银行等。

公司简介

众微在企业信用风险分析、评估及监控方面形成了一系列的专有技术，并以大数据、人工智能等技术为核心，形成具有自主知识产权的平台类产品体“智数-企业全息数据平台”、“智信-大数据企业信用分析平台”和“智融-信用金融服务平台”以及配套的成熟运营与推广方案。在企业信贷领域研发了垂直应用场景的产品，包括企业交易行为分析引擎、财报通等；同时，根据机构用户的应用场景，提供定制化的解决方案。同时，基于核心技术平台建立了垂直行业的解决方案，如面向信贷类的金融机构，方案可以与信贷流程进行充分的融合。

公司发展历程


起步发展
模式升级
迭代推广
2016
2017
2018
2019
2020
研发风控模型产品
参与苏州市金融服务平台体系建设，提供核心风控技术
获得峰谷资本投资
研发信用金融服务平台整体解决方案
迭代模型，并与银行联合研发场景化的算法模型产品
落地河南省焦作市，试点提供平台整体运营服务
试点城市取得近百亿交易额
优化平台与运营方案
落地营口、枣庄、芜湖、北京
获得山东省某地方政府投资
设计不同体量的城市的运营方案
投产平台开始取得运营收入
落地山东、重庆、甘肃、青海、徐州、长春、鞍山、无锡等地
研发场景化风控与信贷产品

公司产品线


智能标签
风险定价模型
信用额度模型
反欺诈模型
风险评估模型（分行业与规模）
行业与区域风险模型
智能推荐算法
智能撮合算法
面向应用的自然语义处理算法
风险动态监控模型组
图数据库
非结构化数据智能处理

AI Lab 简介

- 作为一家初创型企业，我们的仅仅在一些重要的领域会投入重要的资源进行创新；
- 目前我们的团队主要来自于清华北大的博士和少数硕士，共七人；
- 我们合作者主要来自于 DeepMind（一人），Facebook AI（一人），MSRA（三人），字节跳动实验室（一人），第四范式（一人），阿里巴巴平台组（一人），港科大袁会卓副教授（尚未入职），人大严睿教授，华东师范大学吴贤毅教授和莱顿大学 van der Vaart 教授。

AI Lab 主要职责

- 发表论文和比赛；
- 解决两个难题：
  - 中小微企业违约数据不足的问题。
  - 自动特征构建的问题。
- 我们今天主要探讨中小微企业违约数据不足的问题。

大纲

CONTENTS

大纲

金融违约的核心痛点

- 对于中小微企业而言，大部分金融产品很少有违约数据；
- 不同金融产品不能借用；
- 大部分披露的结构化数据很少有信息，例如财报；

解决的可能突破口

- 大部分企业有很多额外的文本信息，例如招聘信息、变更信息、公司简介等。
- 虽然没有违约信息，但是一般的信息是有的。
- 思路：是否可以通过类似于 BERT 预训练的方式提升其效果。

问题重要性

- 结构化数据和文本数据的结合一直是痛点；
- 从信息量来说，除非非常特殊的应用（比如说内容推荐），结构化数据和文本数据信息量是最大的，反倒是 CV 数据量相对较少。
- 结构化数据的预训练仅仅在Arik and Pfister (2019) 中尝试过，其他使用 AutoEncoder 的策略效果微乎其微。
- 很多场景可以应用这种策略，最典型场景是推荐；一些需要高精度分类的也可以采用这种措施。

技术挑战

- 深度学习的表格化数据挖掘不够成熟，Arik and Pfister (2019) 是唯一接近 lightgbm 的（3%）。
- 中文预训练语言模型普遍效果不佳，并且不支持长文本。
- 整体训练过程十分复杂。不进行训练的话，准确度可能会差到 20% 以上。
- 两者结合目前只有很少的探索，但整体效果不佳。
- 单独训练两者并合并效果极差。

其他一些可以使用多模态的场景

- 投诉分类和流失预警：除去投诉文本本身，过去的表现也可以作为流失预警的重要标志；
- 收入分类：出去注释类别外，收入额度也是很重要的判断因素；
- 客户电话的情感分类：不仅仅可以考虑文本本身，还可以考虑音频，加上投诉历史标签

大纲

CONTENTS

大纲

从世界角度 I

- 从整体角度来看，预训练语言模型和 CV 的预训练出现了比较尴尬的局面。
- 一方面来讲，和 CV 不同，不同的预训练模型似乎对提取不同程度的特征有帮助，NLP 的预训练语言模型基本是越大越好（数据集和模型）；
- 另一方面，由于大部分研究者负担不起这么大的算力，只能往以下方向努力
  - 快速训练和推断（fine-tune 阶段）。最早是从 Reformer(Kitaev, Kaiser, and Levskaya 2020) 开始到目前最成熟的 Performer(Choromanski et al. 2020)。这些框架对于**长文本帮助**是很大的。
  - 快速训练。最典型的研究是Clark et al. (2020)，这篇的复现问题导致了很大的争议。

从世界角度 II

- 小模型。从Lan et al. (2019) 开始，各种应接乱七八糟的小模型都出来了，导致效果比较不公平。
- 和知识图谱融合，例如Ostendorff et al. (2019) 等等。但是由于这里面涉及到工作太多，导致结果未必好。

中文角度

- 中文的预训练语言模型可以用惨不忍睹来形容。
- 在中文的比赛中，大部分最终效果最好的，还是谷歌的 BERT 和变体。
- 很多在英文中测试出来的非常好的方法，到了中文和 BERT 的准确性能差 10% 以上，比如说 Albert。
- 大部分人员由于不理解，以追一科技为典型，直接解释为这些研究成果都是造假。

中文语言模型不好的深层原因

- 语料不足，我们以 XLNet(Yang et al. 2019) 为例，Common Crawl 是他主要的训练来源，大小为 PB 级。
- 再以 T5(Raffel et al. 2019) 为例，其除去使用 Common Crawl 外，还使用了大量的任务（主要来自于 GLUE），而对应的中文 CLUE 数据集差的极多。
- 大部分中文预料，最多达到 100G 级别。
- 为了处理这么大的数据，需要大量的算力，究竟多大的算力呢？下面我们来做个计算。

算力问题 I

- XLNet 相对来说耗费算力较少，我们来算一下他的算力耗费。XlNet 仅仅用了 TPU V3-512。而 T5 用的是 TPU V3-2048。我们以后者为例，因为我们通过和 DeepMind 的合作可以比较便宜的拿到这些算力。
- T5 大概训练了一个星期，其公开价格（见TPU 定价），为 147,6608 美元。
- 但是这不是问题，问题在于大部分企业没有办法拿 TPU 训练，也申请不到资源。所以只能拿 GPU 训练。
- 一般来说，TPU V3-8 大概相当于 80 个 V100-32g 的，根据谷歌云定价，其价格为 2 美元一小时（抢占式）。
- TPU V3-2048 相当于 256 块 TPU V3-8 所以，总共需要 20400 块 V100 才能达到其效果。

算力问题 II

- 但是由于分布式训练 V100 会造成自身效率损失，所以至少时间需要延长 4 倍到 8 倍，我们按照 4 倍算，也就是说，需要 80,000 块 V100 一个星期。
- 按照谷歌云价格，其成本为 26,880,000 美元，即约 10 亿人民币。这种投入，任何中国企业不可能能承受。
- 这是假设实验没有失败的情况。一般来说，重新训练五遍是很正常的。
- 对于不相信 TPU 算力差的，我们一会会展示一下 TPU V3-8 的算力。
- 所以中国怎么做呢？大部分只用很小的数据，一遍过。比如说 Albert 只用了 1G 的维基百科语料。
- 目前智源科技和 one-Flow 在联合准备训练超过 GPT3 的模型，即使按照 one-Flow 官方宣传，假设他比 TensorFlow 快 10 倍（这是不可能的），智源也至少需要投入 1 亿人民币，才能作出像样的预训练语言模型。

算力问题 III

- 云从则尝试和知识图谱结合，但是中文知识图谱也很糟糕。
- 百度 ERNIE large 我们尝试过，大概比 BERT 低 10 个百分点。

提升方法

- 算力和数据问题无法绕过。
- 新的网络架构可以减少复杂度。
- 新的网络设计 trick 可以提升效果。
- post-train 方法可以提升。
- 不同程度的 embedding 可以进行拼接。
- 自动化建模 emsemble 可以提升。

新的网络架构：Performer

- 目前复杂度最低的网络；十分适合长文本。
- Performer 结构比较复杂；我们直接看 PyTorch 实现更容易看懂；
- 唯一问题：QR 在 TPU 的速度。
- Layer Share 问题：隔层 Share 可以减少消耗的内存，并且提高效果。

其他设计 trick

- ReZero(Bachlechner et al. 2020)：替代普通的 Residual Connection;
- Summers and Dinneen (2019) 讨论了 Normalization 的作用。
- Liu et al. (2020) 和Chen et al. (2019) 都提出了 Normalization 和 Residual Connection 和 Dropout 的关系。
- Correia, Niculae, and Martins (2019) 可能会起作用，但是需要精细的调整（目前他们的训练方式有问题），而且需要循环用数值求解，所以很危险。
- Mixture of Expert **edus2021switch**。这方面是非常有效的效果。

Post-train 的方法

- Gururangan et al. (2020) 在实际应用中不佳，原因是会破坏掉原始的结构。
- 更好的方法是采用非常少的轮数，但是非常难的问题进行训练。目前我们发现Lewis et al. (2019) 和Clark et al. (2020) 效果很好，Multi-task 很难得到很好的效果。但是 Triplet Mining 则在一些多任务中有很好的效果（尤其是分类任务）。

语言模型拼接

- 目前来看，Transformer 类拼接效果不好。Transformer 类 +ELmo+Word Embedding 效果好一些。
- 全局信息目前捕捉的并不好，最接近的在于Thongtan and Phienthrakul (2019)，但是效果很一般。

拼接后的训练 I

- 首先 finetune 结束后，固定住下面所有的 backbone。
- 在上层，采用So, Liang, and Le (2019) 的搜索架构，但是采用Dong and Yang (2019)，用 EntMax-α 替换掉原来 softmax 方法。搜索差不多可以开始所有 finetune。
- 在最终层，将下面网络直接 concat 进来，前后反向输入，用所有的 pooling methods 后接全联接。注意最终全链接要把 "[cls]" 直接连接过来。训练时候不同曾用不同学习率。
- 在快要达到最高时候，开始进行细调。主要策略是
  - 使用 Lookahead(Zhang et al. 2019)+sgd+momentum, 下不去就减学习率;
  - Label Smoothing + Focal Loss + Contrast Loss;

拼接后的训练 II

- Triplet Loss Mining (Nina, Moody, and Milligan 2019; Chang et al. 2020; Xuan, Stylianou, and Pless 2020)。这种方法中 CV 极多，是最重要的方法。如何构建 triplet 是解决 badcase 的核心。
- Consistency Regularization。主要使用Xie et al. (2019) 和Sohn et al. (2020)。
- Adversarial Training (Goodfellow, Shlens, and Szegedy 2014; Madry et al. 2017; Wong, Rice, and Kolter 2020)。实验效果不大，但是不妨可以尝试。

- 注意所有这些方法，都不需要从头训练。每次有典型提高就存个 checkpoint。
- 在 k-fold 中重点调低的那个 fold。
- 此外，别忘了关键词拼接。

解决方案

- 目前 Performer 尚未训练完毕；预计 5 月份可以训练完毕，开源使用（效果将会远远超过 bert）；
- 以上复杂的调参环节我们将会开发出来一个自动的库。只需要调整部分参数即可训练。
- 仍然存在这调整参数训练量过大的问题。这点我们将会在后面提到。

大纲

表格化数据挖掘方法

- 对于一般问题，深度学习一般被暴打。
- 唯一希望在于 TabNet(Arik and Pfister 2019)，但是问题极大。
- 我们在此基础之上构建了新的优化器，训练方式和网络架构，希望在 3 月份之前能有结果。

大纲

多模态训练方法

- 解决了表格化数据挖掘和语言模型之后，多模态其实反倒简单。
- 多模态一般是联合进行训练的，单独训练拼接效果不好，大部分抄袭 Bert，例如 (Sun et al. 2019)。
- 最近最大的进展是Facebook Hateful Meme Challenge，其中二等奖获得者为我们的 intern。我们请他简单介绍一下比赛的创新点。

大纲

我们利用小微企业的数据，进行了以下实验（只考虑准确率）

- 最普通的逻辑回归，使用经验数据：79.4%
- 使用 LightGBM 等方法进行精调：84.5%
- 使用非常复杂的 Stacking 方法：85.1%
- 使用 TabNet：80.1%
- 使用我们自创的 AdaTabNet 和 Ada-ranger 优化器：84.9%
- 仅仅用 BERT 进行预测：61.2%
- 使用各种语言模型进行非常精细的调整：67.4%
- 直接单独训练并拼接精调语言模型：83.2%
- 使用共同预训练：89.1%

目前并不确定我们新训练的 Performer 究竟能提高多少。

进展

- 大部分内容将会是我们发表论文后进行公布。
- 并且将之做成 PyTorch 一套开源工具。
- 但是还有一个落地问题：这么复杂的调参过程跑得动么？

大纲

CONTENTS

大纲

分布式能不能解决我们的问题

- 大部分厂商都在吹嘘自己的分布式解决方案；
- 但是是不是分布式就能解决问题呢？

分布式的基本假设

- 一般来说，分布式都是通过数据分布实现的；换句话说，我们把数据分散在不同的 GPU 当中，计算梯度并聚合。
- 那么意味着只有 Batch Size 足够大的时候，我们才有可能能用分布式。
- 问题在于，Batch Size 大了以后，对应的优化器必须做改变，否则的话，我们并不能减少训练轮数。可惜目前看起来最有希望的 Lamb(**you2019large**)，实践效果极差。
- 其他优化内部实现，如非常优秀的 DeepSpeed(**rasley2020deepspeed**) 也只能起到微小的作用。

一个更现实的问题

- 训练慢的情况有两种，一种是模型复杂，一种是数据量大；
- 前者没法动，后者的假设常常是如果数据量大，不用调模型就可以解决问题。
- 实际上，目前要调的模型越来越多，精调的模型在极少的数据集上也可以超过极大的数据集。
- 而且很多调整需要从头开始。

一个曲线救国的方法

- 使用 CoLab TPU。
- 一个 TPU 等于 40 张 V100。
- 50 人民币一个月随便用。
- PyTorch 支持更好。
- 虽然不能用数据，但是尝试调参也可以提高效果。
- 我们正在和 PyTorch XLA 合作提升效果。

大纲

TensorFlow 1.x 和 2.x

- 整体来说，TF 1.x 已经几乎不再被维护；除非是 legacy 代码否则没办法处理；
- 而且随着 runtime 的更新，legacy 代码也有可能会被干掉；
- TF 1.x 的有点主要是静态图，TF 2.x 希望在这基础上改成类似于 PyTorch 一样的 API。
- 结果 → 水土不服，bug 成堆，官方代码各种跑不通。

PyTorch

- 希望通过 XLA 解决运行速度问题；
- 一堆 Bug。
- 最大问题，在 TF 中我们可以实现一次将多个东西喂给 TPU，加速训练，因为 TPU 的主要平静瓶颈在于 CPU 的数据传输速度问题。最近 XLA 终于实现了这点，但是代价是每一次都要复制模型。
- 此外，PyTorch Lightning理论上可以支持各种训练情况转移，实际上 bug 极多。

我们希望什么样的框架！

- PyTorch 的易用性；TensorFlow 的速度；
- Async in nature；
- JIT 支持；
- XLA 支持；
- 代码越简单越好。

大纲

- Jax 本身是一个打了鸡血 numpy；换句话说大部分 numpy 的 api 均可以直接使用；
- 常见办法，直接对着 pytorch 代码改，改完就跑。
- Jax numpy 的问题在于他本身是 immutable 的。所以需要使用一定函数式编程的技巧。尽量用lax当中的操作替代。
- vmap 和 pmap可以很容易实现并行。

JIT 和 Jaxpr I

- Jit可以实现 just-in-time compilation。
- 理论上这种方法，可以保证在运行时候运用更多的信息；但是这点在async dispatch当中更有用。Async Dispatch 的原因在于本身来说函数都是没有负作用的，所有自动就可以实现。
- Jit 最有用的两点在于：
  - static_argnums：可以帮助确定 compile time constant，同时可以解决 trace 问题；
  - Jaxpr 的生成可以帮助我们看到实际运算；见官方文档。
- 注意几件事情：
  - static_argnums 必须一致；不同函数中的 static_argnums 必须一致；注意当出现在字典中的时候情况就不一样了；
  - static_argnums 会改变函数参数顺序；

JIT 和 Jaxpr II

- static_argnums 会无法对 PyTree 进行操作；
- 常见工作顺序：
  - 按照 PyTorch 的代码抄实现；
  - 不加 jit 跑通；
  - 自己写backward；这部分往往是最浪费算力的地方；
  - 加 Jit 并跑通；
  - 检查 Jaxpr 看是否满意；
  - 用 profiler 看是否有其他地方要修改。

Primitives

- 在之前我们提过 tracable 的问题；基本意思是只有满足一定条件的数据或者操作才可以进行操作。
- 但这意味着，你其实可以自己控制全部编译过程，甚至在不同的硬件上跑。
- 见Primitives。

PyTree

- PyTree 一般指 nested dict。
- PyTree 自身有非常好的支持。见PyTree定义。

Flax

- Flax **几乎和** PyTorch **长成一样**;
- **例如**这个例子。

Jax 和 Flax 的应用

- 使用 CoLab 的 TPU 对模型进行实验；
- 即使不是真实数据，也可以了解模型的性质；
- CoLab 大概 50 人民币一个月；TPU 随便用。

Jax 和 Flax 的问题

- 生态圈不完全；我将会通过博文视点第一本 Jax/Flax 的书
- Bug 虽然不多，但是文档不全；
- data 处理依托于 TF datasets；
- 需要自己写大量的基础代码；
- 优化器部分设计有问题，导致实现一些操作（例如不同层不同学习率）很困难。
- 没有 Sparse 运算，虽然很容易处理。

- 建议方式：聘用 Jax/Flax 专家重写代码；
- 我们在 GPU 上得到至少 50 倍的加速。

Questions?

大纲

CONTENTS

- Arik, Sercan O and Tomas Pfister (2019). "Tabnet: Attentive interpretable tabular learning". In: *arXiv preprint arXiv:1908.07442*.
- Bachlechner, Thomas et al. (2020). "Rezero is all you need: Fast convergence at large depth". In: *arXiv preprint arXiv:2003.04887*.
- Chang, Xinyuan et al. (2020). "Transductive semi-supervised metric learning for person re-identification". In: *Pattern Recognition* 108, p. 107569.
- Chen, Guangyong et al. (2019). "Rethinking the Usage of Batch Normalization and Dropout in the Training of Deep Neural Networks". In: *arXiv preprint arXiv:1905.05928*.
- Choromanski, Krzysztof et al. (2020). "Rethinking attention with performers". In: *arXiv preprint arXiv:2009.14794*.
- Clark, Kevin et al. (2020). "Electra: Pre-training text encoders as discriminators rather than generators". In: *arXiv preprint arXiv:2003.10555*.

- Correia, Gonçalo M, Vlad Niculae, and André FT Martins (2019). "Adaptively sparse transformers". In: *arXiv preprint arXiv:1909.00015*.
- Dong, Xuanyi and Yi Yang (2019). "Searching for a robust neural architecture in four gpu hours". In: *Proceedings of the IEEE Conference on computer vision and pattern recognition*, pp. 1761–1770.
- Goodfellow, Ian J, Jonathon Shlens, and Christian Szegedy (2014). "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572*.
- Gururangan, Suchin et al. (2020). "Don't Stop Pretraining: Adapt Language Models to Domains and Tasks". In: *arXiv preprint arXiv:2004.10964*.
- Kitaev, Nikita, Łukasz Kaiser, and Anselm Levskaya (2020). "Reformer: The efficient transformer". In: *arXiv preprint arXiv:2001.04451*.
- Lan, Zhenzhong et al. (2019). "Albert: A lite bert for self-supervised learning of language representations". In: *arXiv preprint arXiv:1909.11942*.

- Lewis, Mike et al. (2019). "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension". In: *arXiv preprint arXiv:1910.13461*.
- Liu, Fenglin et al. (2020). "Rethinking Skip Connection with Layer Normalization". In: *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 3586–3598.
- Madry, Aleksander et al. (2017). "Towards deep learning models resistant to adversarial attacks". In: *arXiv preprint arXiv:1706.06083*.
- Nina, Oliver, Jamison Moody, and Clarissa Milligan (2019). "A Decoder-Free Approach for Unsupervised Clustering and Manifold Learning with Random Triplet Mining". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 0–0.
- Ostendorff, Malte et al. (2019). "Enriching BERT with knowledge graph embeddings for document classification". In: *arXiv preprint arXiv:1909.08402*.

- Raffel, Colin et al. (2019). “Exploring the limits of transfer learning with a unified text-to-text transformer”. In: *arXiv preprint arXiv:1910.10683*.
- So, David R, Chen Liang, and Quoc V Le (2019). “The evolved transformer”. In: *arXiv preprint arXiv:1901.11117*.
- Sohn, Kihyuk et al. (2020). “Fixmatch: Simplifying semi-supervised learning with consistency and confidence”. In: *arXiv preprint arXiv:2001.07685*.
- Summers, Cecilia and Michael J Dinneen (2019). “Four Things Everyone Should Know to Improve Batch Normalization”. In: *arXiv preprint arXiv:1906.03548*.
- Sun, Chen et al. (2019). “Videobert: A joint model for video and language representation learning”. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 7464–7473.

- Thongtan, Tan and Tanasanee Phienthrakul (2019). "Sentiment classification using document embeddings trained with cosine similarity". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pp. 407–414.
- Wong, Eric, Leslie Rice, and J Zico Kolter (2020). "Fast is better than free: Revisiting adversarial training". In: *arXiv preprint arXiv:2001.03994*.
- Xie, Qizhe et al. (2019). "Unsupervised data augmentation for consistency training". In: *arXiv preprint arXiv:1904.12848*.
- Xuan, Hong, Abby Stylianou, and Robert Pless (2020). "Improved embeddings with easy positive triplet mining". In: *The IEEE Winter Conference on Applications of Computer Vision*, pp. 2474–2482.
- Yang, Zhilin et al. (2019). "Xlnet: Generalized autoregressive pretraining for language understanding". In: *Advances in neural information processing systems*, pp. 5753–5763.

 Zhang, Michael et al. (2019). “Lookahead optimizer: k steps forward, 1 step back”. In: *Advances in Neural Information Processing Systems*, pp. 9597–9608.

见微知著

联系我们 010-5395 9954 wedo@wecredo.com 中国北京市朝阳区 SOHO 尚都北塔 A 座 1906 100020