知乎专家发现算法竞赛实践报告

黄巧
2019 级计算机技术
学号 201934715

ABSTRACT

知识分享服务已经成为目前全球互联网的重要、最受欢迎的应用类型之一。在知识分享或问答社区中，问题数远远超过有质量的回复数。因此，如何连接知识、专家和用户，增加专家的回答意愿，成为了此类服务的中心课题。

知乎每天有数以十万计的新问题，如何高效的将这些用户新提出的问题邀请其他用户进行解答，以及挖掘用户有能力且感兴趣的问题进行邀请下发，优化邀请回答的准确率，提高问题解答率以及回答生产数，成为知乎最重要的课题之一。

本文所做研究针对上述问题提出了可能的解决方案：根据知乎提供的问题信息、用户画像、用户回答记录，以及用户接受邀请的记录数据，预测这个用户是否会接受某个新问题的邀请。

首先在数据预处理以及特征提取部分，对给出的数据集，解析列表以及重编码，提取出用户特征、问题特征、答案特征，并根据三个数据集之间的关联信息分析进行交叉特征提取；对处理后的数据集进行 StratifiedKFold 分层采样，区分训练集和测试集；在模型训练部分，采用基于 XGBoost 优化的 LightGBM 算法，通过控制树的深度和每个叶子节点的最小数据量，避免过拟合现象，进行模型训练。实验准确率在验证集上为 0.82104，最终测试集上结果为 0.82127。

KEYWORDS

LightGBM, 交叉特征提取, 特征工程, 知乎问题推荐

ACM Reference Format:

黄巧. 2019. 知乎专家发现算法竞赛实践报告. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

1 简介

知乎是中文互联网知名的可信赖问答社区，致力于构建一个人人都可以便捷接入的知识分享网络，让人们便捷地与世界分享知识、经验和见解，发现更大的世界。知乎用户们通过知识建立信任和连接，对热点事件或话题进行理性、深度、多维度的讨论，分享专业、有趣、多元的高质量内容，打造和提升个人品牌价值，发现并获得新机会。

知乎目前已拥有超过 2.2 亿的用户，每天产生海量的提问，传统的手动邀请他人回答问题功能已经不能满足用户的需求，故知乎专家推荐系统即问题路由应运而生，工作机制如图 1 所示。问题路由推荐系统每日可以对 10+ 万的问题进行分发，并保证问题提问后 3 日内的解答率达到百分之七十以上；系统对千万级的创作群体进行精准推荐，经由系统智能分发推荐下的问题，每日产生的回答数超过 20 万。因此，探索高效精准的推荐算法，挖掘有能力且感兴趣的用户进行问题的精准推荐，成为知乎最重要的课题之一。

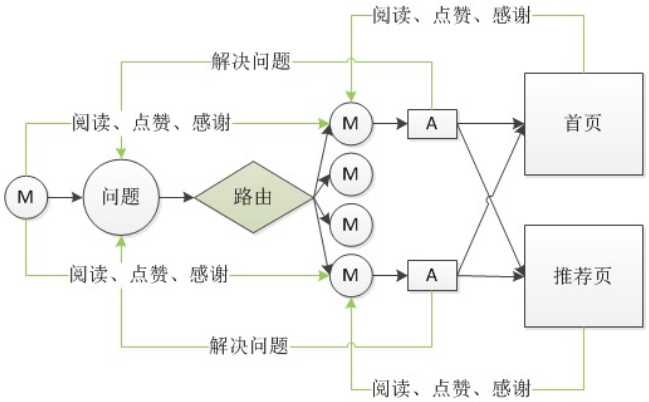

Figure 1: 知乎路由工作机制.

问题路由主要包括排序、召回、特征落地三个模块，如图 2 所示。系统首先根据用户 ID 向问题路由发起请求，通过用户画像获取用户特征（年龄、兴趣）；然后使用召回模块召回用户潜在能够回答的问题，从数百万的问题中找到几百个问题供排序模块使用；接下来通过特征模块获取问题的相关特征，据此对问题进行精准排序，并将头部问题返回至用户，排序中产生落地的特征会记录在日志中和用于训练模型。

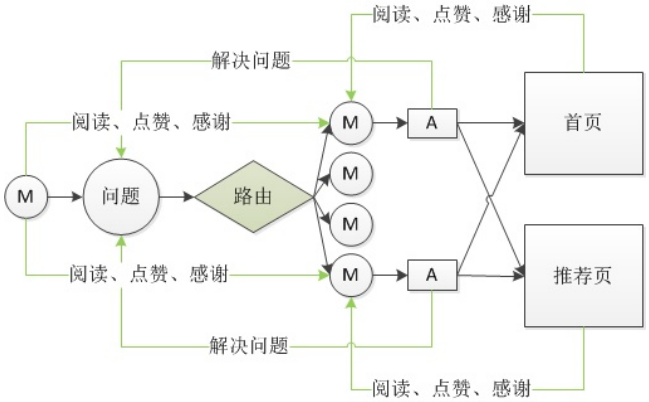

Figure 2: 知乎问题路由内部实践形式.

本文所做研究提出了一种可能的解决方案，根据知乎提供的问题信息、用户画像、用户回答记录，以及用户接受邀请的记录数据，预测这个用户是否会接受某个新问题的邀请。

在数据预处理以及特征提取部分，对给出的数据集，解析列表以及重编码，提取出用户特征、问题特征、答案特征，并根

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

据三个数据集之间的关联信息分析进行交叉特征提取。在问题侧特征方面，由于问题本身是一个 20 个字左右的短文本，可提取的信息有限，所以利用了话题、创建时间等相关信息，在文本信息上使用 embedding。同时，由于问题存在话题特征，回答没有，所以将问题的话题特征作为回答的特征；在用户侧特征方面，包括年龄、性别等基本特征，以及根据用户的历史点击、浏览挖掘的用户兴趣。最重要的特征是用户历史回答特征（时间、长度、字数、赞同数等），通过特征工程提取到大量特征，如总历史回答数、总赞同数、上次回答时间、回答频率、擅长领域等；在交叉特征方面，计算了问题话题和关键词的 embedding 与用户历史回答话题和关键词的 embedding 的相似度。

在模型训练部分，对处理后的数据集进行 StratifiedKFold 分层采样，区分训练集和测试集；在模型训练部分，采用基于 XGBoost 优化的 LightGBM 算法，通过控制树的深度和每个叶子节点的最小数据量, 避免过拟合现象，进行模型训练。实验准确率在验证集上为 0.82104，最终测试集上结果为 0.82127。

2 相关工作

本文主要关注基于集成学习的机器学习方法，集成学习通过构建并结合多个个体学习器来完成学习任务。以分类任务为例，在对新的数据进行分类时，通过训练多个分类器，把这些分类器的分类结果按照不同的组合方式得到最终的分类结果。对于一个复杂任务，将多个专家的判断进行适当的综合所得出的判断，要比其中任何一个专家单独的判断要好得多。通过使用多个个体学习器共同决策一个实例的分类从而提高分类器的泛化能力。

目前来讲，按照个体学习器组合方式的不同，集成学习可分为以下三大方法。

Bagging[1]：通过对原数据集进行有放回的抽取，构建出多个样本数据集，然后用这些新的数据集训练多个分类器。因为是有放回的采用，所以一些样本可能会出现多次，而其他样本会被忽略。该方法通过降低个体学习器方差来改善泛化能力，因此 Bagging 的性能依赖于个体学习器的稳定性，如果个体学习器是不稳定的，Bagging 有助于减低训练数据的随机扰动导致的误差，但是如果个体学习器是稳定的，即对数据变化不敏感，那么 Bagging 方法就得不到性能的提升，甚至会减低。该方法的典型代表为随机森林算法。

Boosting[4]：算法的优化是一个迭代的过程，通过改变样本分布，使得分类器聚集在那些很难分的样本上，对那些容易错分的数据加强学习，增加错分数据的权重，这样错分的数据再下一轮的迭代就有更大的作用，即对错分数据做出惩罚。该方法的典型代表包括 AdaBoost 算法、GBDT 算法等等。

Stacking：即模型融合，整个算法中包含了两个阶段的模型训练和预测流程。在第一阶段，训练拟合多个模型，这些模型既可以属于不同种类，也可以是同一类但有着不同的参数设置，主要目的在于保证模型之间具有一定的差异性。接着对不同模型做出的预测结果进行加权融合，作为第二阶段的训练数据，进行新一轮的模型构建和最终的结果预测。

在 Xile Gao，Haiyong Luo[2] 等人的研究中提出了一种基于 SDAE (叠加去噪自动编码器) 和 LightGBM[3] 的算法。采用 SDAE 对原始数据中的噪声进行处理，并用非监督式学习提取最有效的特征表达式。LGB 揭示了准确识别人类活动类别之间的内在特征依赖关系。在三个典型的应用场景中，对四个数据集进行了广泛的实验，即人的移动模式、当前的静态和用户的动态行为。实验结果表明，该算法的平均准确率达到百分之 95.99，优于其他使用 XGBoost、CNN、CNN+ 统计特征或单个 SDAE 的比较算法。因此，LightGBM 算法在表现上具有更高准确率、更快训练效率、支持直接使用 category 特征等优点。

3 方法

3.1 LabelEncoder 编码

在数据预处理模块，LabelEncoder 编码将离散型变量转换成连续的数值型变量，即对不连续的数字或者文本进行编号，转化为数值。

3.2 K 折交叉验证

交叉验证的基本思想是把在某种意义下将原始数据 (dataset) 进行分组, 一部分做为训练集 (train set), 另一部分做为验证集 (validation set), 首先用训练集对分类器进行训练, 再利用验证集来测试训练得到的模型 (model), 以此来做为评价分类器的性能指标。

K 折交叉验证的原理：假设数据集为 D，将 D 分为 k 个不相交的子集。如果 D 中有 m 个样本，那么每个子集中都有 m/k 个样本，每个子集都尽可能保持数据分布的一致性。每次使用 k-1 个子集作为训练数据，用 1 个子集作为测试数据，训练 k 次。最终的结果是这 k 次测试结果的均值。K 折交叉验证方法的优势在于，同时重复运用随机产生的子样本进行训练和验证，每次的结果验证一次，10 折交叉验证是最常用的。

采用交叉验证的方法可以通过降低模型在一次数据分割中性能表现上的方差来保证模型性能的稳定性，但在数据集很大时，计算过程会相对缓慢。

3.3 LightGBM 算法

LightGBM 算法均属于集成学习领域中的 Boosting 流派, 是一个开源、快速、高效的基于决策树算法的提升框架, 支持高效的并行训练，是在 XGBoost 算法的基础上的优化，具有低内存使用、更高准确率、更快训练效率、支持并行学习、可处理大规模数据、支持直接使用 category 特征的优点。

XGBoost 使用的是 pre-sorted 算法，能够更精确的找到数据分隔点。首先，对所有特征按数值进行预排序；其次，在每次的样本分割时找到每个特征的最优分割点；最后，找到最后的特征以及分割点，将数据分裂成左右两个子节点。这种算法能够准确找到分裂点，但是在空间和时间上有很大的开销。由于需要对特征进行预排序并且需要保存排序后的索引值（为了后续快速的计算分裂点），因此内存需要训练数据的两倍。在遍历每一个分割点的时候，都需要进行分裂增益的计算，消耗的代价大。

LightGBM 使用的是 histogram 算法，占用的内存更低，数据分隔的复杂度更低。其思想是将连续的浮点特征离散成 k 个离散值，并构造宽度为 k 的 Histogram。然后遍历训练数据，统计每个离散值在直方图中的累计统计量。在进行特征选择时，只需要根据直方图的离散值，遍历寻找最优的分割点。

LightGBM 采用按叶子分裂的方法, 计算代价小, 通过控制树的深度和每个叶子节点的最小数据量, 避免过拟合现象，可以减小储存成本和计算成本。另外, 类别特征的处理也使得 LightGBM 在特定数据下有比较好的性能提升。

4 实验分析

4.1 数据预处理

4.1.1 **数据集** *1 single-word-vectors-64d.txt.* 数据分析：每一行代表一个单字及 64 维 embedding 的表示，每一行有 65 列，第一列以 SWxxx 表示单字编码序号, 其后是 64 个浮点数，代表 64 维 embedding 向量。

数据处理：embed 转化为列表，id 转化为 int，生成 single-word.pkl 文件。single-word=['id', 'embed']

4.1.2 **数据集** *2 word-vectors-64d.txt.* 数据分析：每一行代表一个词及其 64 维 embedding 的表示，每一行有 65 列，第一列以 Wxxx 表示词编码序号, 其后是 64 个浮点数，代表 64 维 embedding 向量。

数据处理：embed 转化为列表，id 转化为 int，生成 word.pkl 文件。word=['id', 'embed']

4.1.3 **数据集** *3 topic-vectors-64d.txt.* 数据分析：每一行代表一个话题及其 64 维 embedding 的表示，每一行有 65 列，第一列以 Txxx 表示话题 ID 编码序号，其后是 64 个浮点数，代表 64 维 embedding 向量。

数据处理：embed 转化为列表，id 转化为 int，生成 topic.pkl 文件。topic=['id', 'embed']

4.1.4 **数据集** *4 question-info-0926.txt.* 数据分析：包含邀请数据集 (数据集 7 和 8) 及回答数据集 (数据集 5) 涉及的所有问题列表，每一行代表一个问题的相关信息, 每一行有 7 列。数据格式如下：[问题 ID 问题创建时间问题标题的单字编码序列问题标题的切词编码序列问题描述的单字编码序列问题描述的词编码序列问题绑定的话题 ID]

数据处理：获取问题创建时间 day-hour, 将 3-7 列转化为列表，生成 question-info.pkl 文件。

question-info=['qid', 'title-t1', 'title-t2', 'desc-t1', 'desc-t2', 'topic', '$q\text{-}day$', '$q\text{-}hour$']

4.1.5 **数据集** *5 answer-info-0926.txt.* 数据分析：包含邀请数据集 (数据集 7 和 8) 中用户最近 2 个月内的所有回答，每一行代表一个回答的相关信息, 每一行有 20 列。数据格式如下：[回答 ID 问题 ID 用户 ID 回答创建时间回答内容的单字编码序列回答内容的切词编码序列回答是否被标优回答是否被推荐回答是否被收入圆桌是否包含图片是否包含视频回答字数点赞数取赞数评论数收藏数感谢数举报数没有帮助数反对数]

数据处理：获取回答创建时间 day-hour，将单字编码以及切词编码转化为列表，生成 answer-info.pkl 文件。

answer-info=['aid', 'qid', 'uid', 'ans-t1', 'ans-t2', 'is-good', 'is-rec', 'is-dest', 'has-img', 'has-video', 'word-count', 'reci-cheer', 'reci-uncheer', 'reci-comment', 'reci-mark','reci-tks', 'reci-xxx', 'reci-no-help', 'reci-dis', 'a-day', 'a-hour']

4.1.6 **数据集** *6 member-info-0926.txt.* 数据分析：包含邀请数据集 (数据集 7 和 8) 中用户相关特征信息，每一行代表一个用户的相关信息，每一行有 21 列。数据格式如下：[用户 ID 性别创作关键词的编码序列创作数量等级创作热度等级注册类型注册平台访问频率用户二分类特征 A B C D E(0/1) 用户分类特征 A B C D E 用户盐值分数用户关注话题用户感兴趣话题]

user=['uid', 'gender', 'creat-keyword', 'level', 'hot', 'reg-type', 'reg-plat', 'freq', 'uf-b1', 'uf-b2','uf-b3','uf-b4','uf-b5','uf-c1','uf-c2','uf-c3','uf-c4','uf-c5','score','follow-topic','inter-topic']

数据处理：查看 user 的 21 列特征在数据集中的唯一值个数：对于 ['creat-keyword', 'level', 'hot', 'reg-type', 'reg-plat'] 在数据集中都仅有 1 个取值，没有参考价值予以删除；['gender', 'freq', 'uf-c1', 'uf-c2', 'uf-c3', 'uf-c4', 'uf-c5'] 为不连续的文本，根据 LabelEncoder 编码方法转化为连续的数值型变量；['follow-topic','inter-topic'] 转化为列表形式；最后生成 answer-info.pkl 文件。处理后的 user 包含 16 个特征：user=['uid','gender','freq','uf-b1','uf-b2','uf-b3','uf-b4','uf-b5','uf-c1','uf-c2','uf-c3','uf-c4', 'uf-c5', 'score', 'follow-topic', 'inter-topic']

4.1.7 **数据集** *7 invite-info-0926.txt.* 数据分析：包含用户最近 1 个月的邀请数据，每一行代表一个问题邀请的相关信息，每一行有 4 列，[问题 ID 用户 ID 邀请时间是否被回答 (0/1)]

数据处理：转换邀请时间格式 day-hour，生成 train.pkl 文件。train=['qid', 'uid', 'label', 'day', '$hour$']

4.1.8 **数据集** *8 invite-info-evaluate-1-0926.txt.* 数据分析：未来 7 天的问题邀请数据, 每一行代表一个问题邀请相关信息, 每一行有 3 列，[问题 ID 用户 ID 邀请时间]

数据处理：转换邀请时间格式 day-hour，生成 test.pkl 文件。test=['qid', 'uid', 'day', '$hour$']

4.2 特征工程

特征工程是将原始数据转化为特征的过程，这些特征可以更好地向预测模型描述潜在问题，从而提高模型对未见数据的准确性，包含了特征提取、特征构建、特征选择等模块。

4.2.1 **特征处理与提取**. 将 user、question、answer 中的字、词、话题特征，与 single-word、word、topic64 维 embedding 向量结合，得到对标题、描述、关键词、话题的特征处理如下：

1、topic.pkl+ user.pkl 提取用户关注及感兴趣话题，写入 user-feat.pkl 文件中

读取 topic.pkl 文件并转化为字典形式 topic-vector-dict。读取 user.pkl 文件有 16 个特征，其中用户关注的话题 follow-topic 和用户感兴趣的话题 inter-topic 为 64 维 embedding，计算话题向量的平均值 user['follow-topic-vector','inter-topic-vector']，分别包括 64 个特征，用户话题特征一共 146 个，存储在 user-feat.pkl 文件中。

2、word.pkl+member-info-0926.txt 提取用户创作关键词，写入 user-keyword-feat.pkl 文件中

读取 word.pkl 文件并转化为字典形式 word-vector-dict, 读取 member-info-0926.txt 文件有 21 个特征，其中 user['creat-keyword'] 为 64 维 embedding，计算词向量均值，添加 user['keyword-vector'] 特征，包括 64 个特征，再加上用户标识 uid 一共 65 个用户创作关键词特征，存储在 user-keyword-feat.pkl 文件中。

3、single-word.pkl + word.pkl+topic.pkl+question-info.pkl，提取问题的标题、描述、话题特征，写入 ques-feat.pkl 文件中

读取 question-info.pkl 有 8 个问题特征，question-info=['qid', 'title-t1', 'title-t2', 'desc-t1', 'desc-t2', 'topic', '$q\text{-}day$', '$q\text{-}hour$']，其中 ['title-t1', 'title-t2', 'desc-t1', 'desc-t2','topic'] 均为 64 维 embedding 向量，分别计算五个特征的话题向量的平均值 ques['$title\text{-}sw\text{-}vector$','$title\text{-}w\text{-}vector$','$desc\text{-}sw\text{-}vector$','$desc\text{-}w\text{-}vector$','topic-vector']，分别细分有 64 个特征，共有 333 个特征，存储在 ques-feat.pkl 文件中。

4、question+answer+user，根据主题合并，得到问题答案用户交叉特征。

将问题信息 question-info.pkl(保留 qid+topic) 和答案信息 answer-info.pkl(保留 qid+uid) 按照 qid 合并，得到问题和答案关于主

题 topic 的联合特征 [‘qid’ ,’ uid’ ,’ topic’]，并计算话题向量的平均值 [’topic-vector’]，包括有 64 个特征。再联合用户信息 user-feat.pkl(保留 uid+inter-topic-vector)，联合用户感兴趣的主题，得到问题答案用户三者交叉特征。answer-q-topic-vector =[‘uid’ , ’ inter-topic-vector’ , ’ qid’ , ’ topic’ , ’ topic-vector’]

4.2.2 **特征选择与构建**. 问题的热度以及时效性对于问题是否会被回答具有较大的影响，因此，结合验证集是对未来 7 天的数据预测，将训练集和验证集进行 7 天的错位，并对特征的选取进行时间限定。分别对通过时间限定之后的数据集中的用户特征、问题特征、答案特征进行统计分析，根据用户 ID 分组计算其均值 mean、计数 count、总数 sum、最值 max 等，用到的主要特征包括有：

特征	特征说明
'gender', 'freq', 'uf_b1', 'uf_b2'...	用户数据集原始特征
'num_follow_topic', 'num_interest_topic'	用户关注和感兴趣的topic数
'most_interest_topic'	用户最感兴趣的topic
'u_inv_mean', 'u_inv_sum', 'u_inv_std', 'u_inv_count'	用户topic兴趣值的统计特征
'keyword_vector'	用户创作的关键词特征

Figure 3: 用户特征.

特征	特征说明
'title_sw_vector', 'title_w_vector' , 'desc_sw_vector, 'desc_w_vector', 'topic_vector'	问题数据集原始特征 标题的单字编码/切词编码 描述的单字编码/词编码序列 问题话题编码
'num_title_sw', 'num_title_w'	标题 词计数
'num_desc_sw', 'num_desc_w'	描述 词计数
'num_qtopic'	topic计数

Figure 4: 问题特征.

特征	特征说明
'num_topic_attent_intersection'	关注topic交集计数
'num_topic_interest_intersection'	兴趣topic交集计数
'min_topic_interest...', 'max...', 'std...', 'mean...'	交集topic兴趣值统计

Figure 5: 交叉特征.

特征	特征说明
'ans_t1', 'ans_t2', 'is_good', 'is_rec', 'is_dest'.....	答案数据集原始特征

Figure 6: 答案特征.

4.3 交叉验证

采取 StratifiedKFold 分层采样，确保训练集、测试集中各类别样本的比例与原始数据集中相同，根据标签中不同类别占比来拆分数据，获取训练集和测试集。

4.4 LightGBM 模型

在模型训练方面，考虑到实际业务场景对运行时间有较高要求（如 500 毫秒），很难使用多个模型融合或者复杂模型，所以倾向于单一模型、多目标多任务模型等，期望模型可解释性强、训练时间短、相对稳定。

本文研究采用 LightGBM 分类模型，采用按叶子分裂的方法，计算代价小，通过控制树的深度和每个叶子节点的最小数据量，避免过拟合现象。训练设置主要参数包括：基学习器模型算法选用传统的梯度提升决策树 GBDT，控制树模型复杂度为 64，学习率 0.01 等。经过实验分析，在迭代次数 3000 时效果最好，在验证集上效果为 0.82104，最终测试集上结果为 0.82127，迭代次数再增加时会出现过拟合现象。

5 结论

本文所做研究针对知乎的邀请回答机制提出了可能的解决方案，根据提供知乎的问题信息、用户画像、用户回答记录，以及用户接受邀请的记录，预测这个用户是否会接受某个新问题的邀请。在数据预处理以及特征提取部分，提取出用户侧特征、问题侧特征、答案侧特征，并根据三个数据集之间的关联信息分析进行交叉特征提取；对处理后的数据集进行 StratifiedKFold 分层采样，并采用基于 XGBoost 算法优化的 LightGBM 算法，实验准确率在验证集上为 0.82104，最终测试集上结果为 0.82127。

实验特征提取部分重点关注了问题话题和用户关注以及感兴趣的话题 topic 特征，结合数据集 1-3 对 single-word、word、topic 的编码处理，进行了交叉特征提取和特征统计，在模型准确率上有一定的效果提升。但对数据集中其他特征如数据集 6 中“用户创作关键词”等的关注度相对较少，后续工作可以增加对其它有效特征的提取和应用，以提升模型训练效果。

REFERENCES

[1] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[2] Xile Gao, Haiyong Luo, Qu Wang, Fang Zhao, Langlang Ye, and Yuexia Zhang. A human activity recognition algorithm based on stacking denoising autoencoder and lightgbm. *Sensors*, 19(4):947, 2019.

[3] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154, 2017.

[4] Robert E Schapire and Yoav Freund. Boosting: Foundations and algorithms. *Kybernetes*, 2013.