基于语义嵌入模型与交易信息的智能合约自动分类系统

黄步添 [1, 2] 刘 琦 [3] 何钦铭 [1] 刘振广 [3] 陈建海 [1]

摘 要 作为区块链技术的一个突破性扩展, 智能合约允许用户在区块链上实现个性化的代码逻辑从而使得区块链技术更加的简单易用. 在智能合约代码信息迅速增长的背景下, 如何管理和组织海量智能合约代码变得更具挑战性. 基于人工智能技术的代码分类系统能根据代码的文本信息自动分门别类, 从而更好地帮助人们管理和组织代码的信息. 本文以 Ethereum 平台上的智能合约为例, 鉴于词嵌入模型可以捕获代码的语义信息, 提出一种基于词嵌入模型的智能合约分类系统. 另外, 每一个智能合约都关联着一系列交易, 我们又通过智能合约的交易信息来更深入地了解智能合约的逻辑行为. 据我们所知, 本文是对智能合约代码自动分类问题的首次研究尝试. 测试结果显示该系统具有较为令人满意的分类性能.

关键词 智能合约, 代码, 交易信息, 词嵌入模型, 神经网络, 长短时记忆模型

引用格式 黄步添, 刘琦, 何钦铭, 刘振广, 陈建海. 基于语义嵌入模型与交易信息的智能合约自动分类系统. 自动化学报, 2017, **43**(9): 1532−1543

DOI 10.16383/j.aas.2017.c160655

Towards Automatic Smart-contract Codes Classification by Means of Word Embedding Model and Transaction Information

HUANG Bu-Tian[1, 2] LIU Qi[3] HE Qin-Ming[1] LIU Zhen-Guang[3] CHEN Jian-Hai[1]

Abstract As an innovative extension of the blockchain technology, smart contract enables users to implement personalized logic. As such, blockchain technology becomes more simple and useful. However, due to the rapid increase of the amount of smart contract codes, managing smart contract codes is becoming much more challenging. Automatic code classifier, which rests on the machine learning methods, can automatically identify the categories of the codes so as to saves a lot of human efforts. In this paper we investigate the smart contract codes of the Ethereum platform and propose a novel smart contract code classifier. To the best of our knowledge, this is the first exploration on automatic classification of the smart contract codes. The classifier is based on the word embedding model. Since each smart contract corresponds to a series of transactions, we further utilize the transactions in the contract to understand the intrinsic logic of the contract. Extensive experiments have verified the effectiveness of our proposed system.

Key words Smart contract, codes, transaction information, word embedding, neural network, long-short term memory

Citation Huang Bu-Tian, Liu Qi, He Qin-Ming, Liu Zhen-Guang, Chen Jian-Hai. Towards automatic smart-contract codes classification by means of word embedding model and transaction information. *Acta Automatica Sinica*, 2017, **43**(9): 1532−1543

随着比特币等加密货币的日益普及, 作为一种创新性的去中心化框架与分布式计算范式, 区块链技术迅速崛起并应用到了众多领域. 其中, 区块链技术在彩票、债券、医疗等领域的应用尤其表现出了令人振奋的前景. 区块链技术利用 POW (Proof of work)[1]、PBFT (Practical byzantine fault tolerance)[2] 等算法在去中心化的环境下达成共识, 共同维护一个分布式的账本, 部分解决了拜占庭将军问题.

在比特币中, 为了实现验证公钥、签名等应用, 比特币实现了一种基于栈的脚本编程语言. 其中, 最常见的脚本是付款给某一个公钥地址. 值得注意的是, 比特币的脚本语言只支持 256 种语言命令. 而且, 在这 256 个命令中, 有 15 个被禁用, 75 个被保留. 因此, 比特币的脚本语言能实现的功能有一定的局限性. 总结而言, 比特币的脚本语言的缺陷主要有以下三点.

1) 图灵完备问题: 有许多运算是比特币的脚本语言不支持的. 比如循环, 为了安全性和防止无限循环, 比特币脚本语言禁用了循环语句.

2) 区块链数据缺失: 在脚本语言中, 程序可以获

收稿日期 2016-09-14 录用日期 2017-02-03
Manuscript received September 14, 2016; accepted February 3, 2017
本文责任编委 袁勇
Recommended by Associate Editor YUAN Yong
1. 浙江大学计算机科学与技术学院 杭州 310007 中国 2. 杭州云象网络技术有限公司 杭州 310012 中国 3. 新加坡国立大学计算机学院 新加坡 119613 新加坡
1. College of Computer Science and Technology, Zhejiang University, Hangzhou 310007, China 2. Yunxiang Network Corporation Hangzhou 310012, China 3. National University of Singapore Singapore 119613, Singapore

得公钥、签名等用户信息. 但是并不能获取区块链块号、区块链哈希等区块链的信息. 因为许多应用比如彩票、博彩需要这些信息来实现随机性, 这限制了脚本语言能够实现的应用范围.

3) 状态缺失: 在付款中, 脚本语言只能实现付款成功或失败两种状态. 因此, 它并不支持更复杂的比如退款、纠纷处理等应用.

4) 客户端不兼容: 随着比特币的发展与安全因素, 越来越多的客户端只支持几种较常见的脚本, 比如 scriptSig 与 scriptPubKey. 导致不常见的脚本无法被执行.

在这样的背景下, Ethereum[1] 等智能合约平台扩展了比特币的脚本语言并实现了一种图灵完备的编程语言. 用户可以利用 Ethereum 提供的编程语言与编程环境来实现个性化的智能合约. 以下是智能合约的一个简单例子:

```
contract SimpleStorage {
    uint s toredData;
    function set (uint x) {
        storedData = x;
    }
     function get ( ) returns (uint retVal)    {
    }
}
```

该例为用 Ethereum 的编程语言 Solidity 所实现的一个智能合约. Ethereum 有两种类型的账号, 一种是普通的用户账号, 一种是智能合约对应的账号. 每一个账号对应着一个地址来指代这个账户. 普通的用户账号记录了用户所有的货币金额 (Ethereum 中记录货币 Ether 的数量), 智能合约对应的账号存储着智能合约的代码和数据. 用户的账号可以发起交易来创建一个智能合约账号并且调用智能合约的函数. 本例中实现了简单的数据存储读取功能, 在将本代码部署到一个智能合约的账号后, 用户可以通过发起交易来修改数据 *storedData* 的内容. 接收到交易后, 区块链在所有节点上运行智能合约的代码, 代码运行在 Ethereum 虚拟机 EVM 上, 最终将 *storedData* 内容更新. 用户已在 Ethereum 上实现了投票、拍卖、投资、游戏、社交网络等多种多样的应用. 截止目前, 有超过 200 种智能合约的应用运行在 Ethereum 的区块链上. 智能合约的地址有超过 118 000[2] 个.

随着 Ethereum 等智能合约平台的迅速发展, 区块链上智能合约代码信息快速增加. 如何管理和组织智能合约代码变得更具挑战性. 用户需要从多种多样的智能合约中选择自己需要的服务. 庞大的智能合约数量使得人工地去选择智能合约变得不再可能. 代码分类系统能够根据代码的文本信息自动分门别类, 避免人工标记海量智能合约的巨大时间成本, 有利于合约代码的分类组织与管理. 然而, 传统的算法, 比如最近邻分类器[3]、朴素贝叶斯分类[4]、决策树[5] 等文本分类算法基于人工定制的特征来实现代码分类, 这样的算法存在两个问题: 1) 人工定制的特征不能实现跨领域的分类, 在一个领域中选择的特征不能应用于另一个领域; 2) 人工定制的特征比如 TF-IDF (Term frequency-inverse document frequency) 存在着特征稀疏的问题, 特征稀疏影响了分类效果的提升.

以词嵌入模型[6] 为代表的神经网络算法有效地解决了上述两个问题. 词嵌入模型可以避免人工选择特征的困难, 可以从文本的上下文中提取出有效的特征, 这些特征是稠密的, 包含了词的语义信息. 并且已经在机器翻译[7]、关系抽取[8] 等领域中取得了超出其他模型的性能. 鉴于词嵌入模型的优良性能, 本文提出一种基于词嵌入模型的智能合约代码分类系统, 根据智能合约的代码信息, 我们将代码中的每一个词都映射成语义空间的向量. 词向量表现了词上下文的本地特征, 为了捕获代码的全局特征, 我们将词向量输入到长短时记忆网络中, 并最终生成一个向量来代表整个代码. 同时, 每一个智能合约地址都关联着一系列的交易, 这些交易揭示了智能合约的逻辑行为. 通过组合代码语义向量与关联的交易信息, 我们的系统可以自动将代码归类为投资、游戏、社交、系统应用等类别. 为了衡量系统的性能, 我们收集了 Ethereum 上的智能合约, 并用 Precision, Recall, F1 等通用指标上与朴素贝叶斯和支持向量机两种基准算法比较, 测试结果显示该系统取得了明显优于朴素贝叶斯和支持向量机性能的效果. 另外, 我们利用本系统实现了一个智能合约分类浏览器和自动检索系统, 方便了用户获取有用的智能合约.

本文的贡献可总结为以下三点:

1) 本文首次提出了用自动代码分类来对智能合约进行管理, 通过对代码分类, 用户可以快速地找到自己需要的服务.

2) 我们提出了一种基于词嵌入模型和智能合约交易信息的自动分类系统.

3) 为了验证系统的性能, 我们收集了 Ethereum 上的智能合约, 并标记了代码的类别. 实验结果显示系统取得了超出基准算法的分类效果

本文的组织结构安排如下: 我们首先在第 1 节中介绍本文相关的工作; 其次, 在第 2 节引出本文的

[1] https://ethereum.org/
[2] https://etherscan.io/

问题; 然后, 在第 3 节和第 4 节中介绍我们的系统架构和实现方法; 最后, 在第 5 节中对本文工作进行总结并展望未来的工作.

1 相关工作

区块链技术将交易记录在公开账本, 所有交易是公开的并且不可逆. 随着区块链技术的发展, 研究者们开展了一系列关于区块链的数据分析方向的研究.

文献 [9] 分析了比特币区块链的公开账本, 通过将公钥地址跟某比特币论坛账号信息连接起来, 继而对交易图上的公钥地址进行聚类. 通过聚类, 作者发现了 Wikileak、FBI 等用户的公钥地址, 并得出结论比特币并不是一个匿名的交易系统. 文献 [10] 量化分析了比特币的交易图, 发现比特币中所有的大型交易都跟 2010 年 11 月的一笔交易有密不可分的关系. 作者同时分析了比特币用户如何通过转账来实现匿名, 作者发现比特币用户有 5 种常用的实现匿名的交易方式. 文献 [11] 用贝叶斯回归来预测加密货币的价格, 并部署系统到实际应用中, 在 60 天的交易过程中取得了收益翻倍的效果. 文献 [12] 通过分析总结智能合约面临的安全问题, 提出了时间依赖、顺序依赖与异常处理三种常见的代码问题. 通过实验发现相当多的智能合约代码存在上述问题, 并用符号执行来识别三种问题, 在实验数据集上取得了较好的预测准确率. 文献 [13] 分析了比特币交易中介的风险. 作者跟踪了 40 个比特币交易平台, 发现 18 个已经关闭. 鉴于比特币交易中介的风险性, 作者用风险模型来提前预测比特币交易平台的风险. 文献 [14] 分析了 Ethereum 智能合约间的调用关系, 并对智能合约聚类. 文献 [15] 可视化了区块链上的交易记录. 每一个节点代表一个记录, 每一条边表示比特币从一条记录流向另一条记录. 通过这样的分析, 可以挖掘交易之间的关联关系.

本文提出的智能合约分类管理问题跟文本分类问题密切相关. 因为超过 80 % 的信息是以文本形式保存的, 文本分类在 Internet 信息处理方面有广泛的应用并有巨大的商业潜在价值. 文本分类将文档分类成预先定义好的类别. 文本分类一般包含以下几步:

1) 预处理: 分词、去无用词 (Stopwords)、词性标注、句子检测等.

2) 特征提取: 提取词特征、语法特征、语义特征等.

3) 预测: 用分类器进行预测.

4) 模型评估: 通常用召回率 (Recall), 精确度 (Precision) 等进行模型评估.

文献 [16] 综述了文本分类的进展. 截止目前, 有相当多的以统计机器学习为基础的文本分类算法被提出. 其中, 典型的有 Rocchio 分类器[17]、最近邻分类器[3]、朴素贝叶斯分类[4]、决策树[5]、生成概率模型[18]、最大熵模型[19] 与向量空间模型 (Vector space model, VSM)[20]. 以上模型需要已标记的训练数据集来训练学习, 并在检验集上测试算法的性能.

向量空间模型是一种有效的文本分类模型, 并且被广泛应用于信息检索、排序等. 向量空间模型通过分词将文档表示成词向量并用 TF-IDF 等加权机制来衡量词权重. 但是, 向量空间模型无法处理一词多义、同义词等情况. 为了解决向量空间模型的问题, 许多作者改进了 TF-IDF 的加权机制, 比如文献 [21−22]. 但是, 这些算法仍然存在特征提取不能跨领域, 特征稀疏等问题.

随着神经网络技术的发展, 深度神经网络[23] 成为一种有效、通用地提取特征与进行预测的模型. 神经网络模型也被应用到文本分类系统. 文献 [24] 提出了一种基于多层次神经网络模型的文本分类系统. 文献 [25] 提出了一种新的分类词典的构建方法并用神经网络来进行文本分类预测.

文本的语义信息可以通过分布式表示方法来建模. 分布式表示方法可以自动提取文本特征, 特征是稠密的, 并且可以有效地表示词的语义. 通过神经网络来学习分布式的表示方法首先由文献 [26] 提出. 谷歌提出了一种词跟短语的分布式表示方法: Word2Vec[6]. Word2Vec 通过 Skip-gram 模型来利用词的上下文信息并且用负采样、层次化的 Softmax 回归来加快训练速度. 文献 [27] 用双线性回归模型来组合潜在语义分析与 Word2Vec. 利用词与词的关联矩阵, 算法可以有效地生成词的语义向量. 基于这些模型, 每一个词或者短语可以表示成一个语义向量. 向量的余弦相似性可以反映词与词之间的相似性. 以 Word2Vec 为基础, 文献 [28] 进一步将文档也表示成一个语义向量并且在情感分析数据集上取得了较好的结果.

基于以上这些工作, 在本文中, 我们提出一种基于词嵌入模型与交易信息的智能合约代码分类系统. 基于代码托管平台 Github 与 Ethereum 区块链浏览器[3], 系统在人工标记的数据集上取得了超出传统算法的分类性能.

2 问题定义

输入一个智能合约的数据集 $\{D_i, y_i\}$, 其中 D_i 为一个智能合约, $y_i \in \{C_1, C_2, \cdots\}$. $\{C_1, C_2, \cdots\}$ 为预定义好的类标签集合. 目标是学习一个映射函

[3] https://etherchain.org/

数 f, 该映射函数能够将输入的一个智能合约 D_i, 映射到它所对应的正确的类标签 y_i. 本文用到的智能合约取自 Ethereum 区块链.

这个问题的挑战主要在于以下几个方面:

1) 因为缺少标记数据集, 如何来标记智能合约的类标签是一个挑战, 人工阅读代码并标记非常消耗时间.

2) 智能合约分类区别于一般的代码分类. 智能合约是对现实生活中各种具有复杂逻辑的需求的抽象. 在 Ethereum 中, 每个智能合约被当做一个账户. 图 1 描述了 Ethereum 区块链的结构, 每个区块用哈希的方式链接成链, 每一个区块有一个 State Root 域, 用户的账号与智能合约的账号用 Merkle 树的形式存储起来. 智能合约账户有它的余额、相关的交易、代码、存储等. 因此, 只利用智能合约的代码信息将丢失关于智能合约的重要信息.

3) 智能合约的代码之间可以互相调用, 互相调用的代码之间在功能上有协同作用, 因此它们的类别也有一定的相似性, 如何解析智能合约代码之间的关系是一个挑战.

鉴于以上这些挑战, 在本文中, 我们将利用词嵌入模型与相关的交易信息来预测智能合约的类别. 在下一节中, 我们将给出系统的框架.

3　系统框架

针对智能合约包含代码与其他交易信息, 我们通过结合智能合约的代码与交易信息来达到更好的分类效果.

如系统框架图 2 所示, 输入一个智能合约, 为了解决特征稀疏的问题, 我们首先将智能合约当中的每一个词映射成一个词嵌入向量. 为了得到代码的全局信息, 我们采用了长短时记忆网络来生成代码的向量表示. 我们将词嵌入向量顺序输入到长短时记忆网络当中, 长短时记忆网络可以有效地记忆词的前文信息, 组合前文和当前词的信息, 最终生成一个全局的代码向量表示.

经过以上步骤之后, 智能合约的代码被转换成了一个语义向量 V. 合约代码相关的交易蕴含了智能合约的行为信息. 比如一个彩票的智能合约, 参与者通过智能合约的押注函数来下注, 押注的交易将包含着押注者转账的信息等. 彩票结束后, 智能合约会发起交易将奖金转账给获奖者. 因此, 智能合约相关的交易可以反映智能合约的内部逻辑. 通过分析智能合约关联交易的信息, 我们提取出交易特征向量 $T = \{t_1, t_2, \cdots\}$.

我们为了组合特征向量 T 和代码的语义向量 V 的信息, 首先将两个连接成一个向量, 然后将这个向量输入到前馈神经网络中 (Feedforward neural network, FNN). 为了捕获非线性的特征关系, 我们设置了一层隐藏层, 最终由 Softmax 层输出每个类标签 $\{C_1, C_2, \cdots\}$ 的概率, 所有的概率相加为一. 我们取其中概率最大的类标签作为这个智能合约的标签.

在下一节中, 我们将详细介绍系统框架的各个部分.

4　算法过程

4.1　代码预处理

在本文中, 我们采用代码的源码而不是编译后的二进制代码形式, 原因主要有:

1) 源码相比较二进制代码有更好的可读性、可解释性、更易于分出类别.

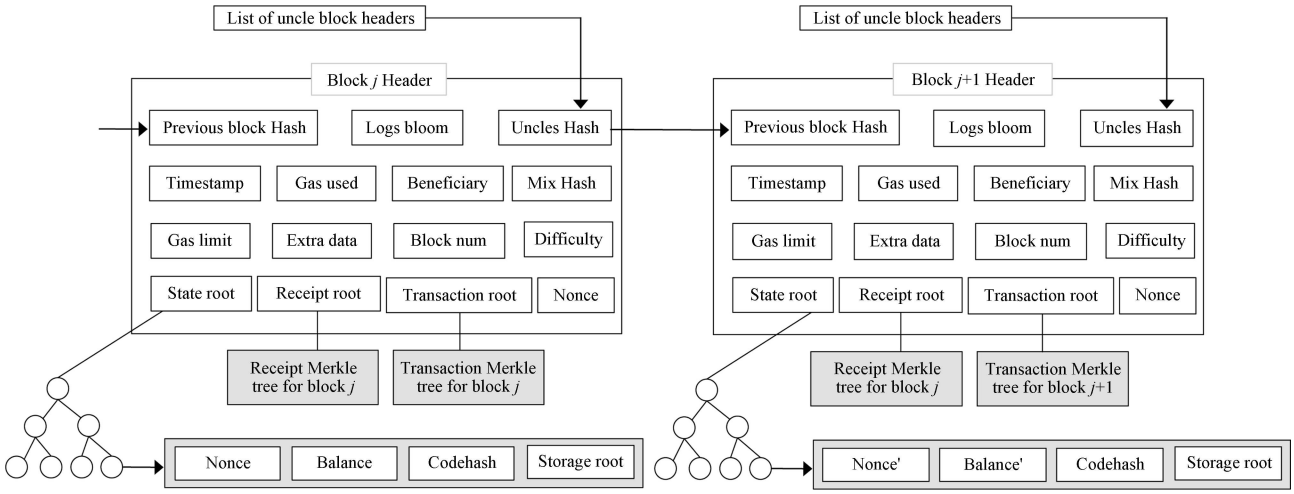


图 1　Ethereum 区块链

Fig. 1　Ethereum blockchain

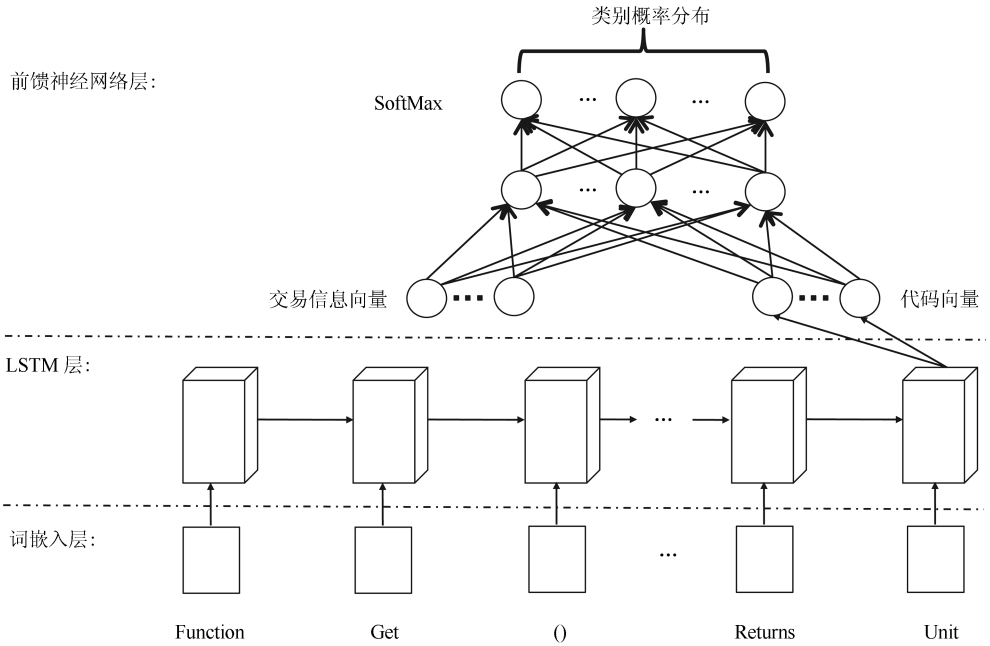


图 2 系统框架

Fig. 2 System architecture

2) 源码有更丰富的表示能力, 如数据类型、异常处理、访问控制等.

3) 源码当中包含着变量名等信息, 变量名反应了智能合约功能的信息, 比如关于彩票的智能合约可能用 'Lottery' 作为名字, 转换成二进制之后变量名信息会丢失.

在得到智能合约的源码后, 因为智能合约之间可以互相调用, 一个智能合约的功能需要另一个智能合约来配合完成, 因此有调用关系的智能合约之间功能上有一定的联系. 为了捕获智能合约间的联系, 在预处理的过程中, 如果一个智能合约调用了另一个智能合约的函数, 我们将被调用的函数扩充到调用函数的对应位置. 在将代码分词后, 代码的每一个词被顺序地输入到神经网络中.

4.2 代码语义向量生成

在预处理智能合约的源代码后, 我们将通过神经网络生成代码的语义向量. 语义向量是理解文本和进行信息检索的一种重要手段, 一个好的模型不应该只捕获表面的字形与句子语法结构, 而且应能够发现句子底层的语义相似性. 神经网络是一种非常好的向量到向量映射的方式, 可以灵活地控制输出向量的大小. 因此我们选择用神经网络来生成智能合约代码的语义向量.

对每一份代码, 我们可以将它看成一个词的序列 $\{w_1, w_2, \cdots\}$. 首先在词嵌入层中, 我们将每一个词 w_i 映射成一个稠密的 300 维的词嵌入向量. 然后将这些词嵌入向量顺序输入到长短时记忆模型中.

长短时记忆模型 (Long short term memory, LSTM)[29] 在自然语言处理中取得了突破性的进展, 比如机器翻译[30], 推理限定继承关系[31] 等. 在机器翻译[30] 中, LSTM 可以将英文句子编码成包含着语义的向量, 然后另一个 LSTM 可以将这个语义向量解码到要翻译的语言, 比如中文. 在文献 [31] 中, 作者用两个 LSTM 来推断限定继承关系. 表达前提的句子被编码成一个语义向量, 假设句子以前提句子的语义向量为输入, 判断两个句子 (前提句子、假设句子) 是否有限定继承的关系. 长短时记忆模型被证明可以捕获输入的长时记忆, 因此可以用来发现输入的内部结构与依赖关系. 并且长短时记忆模型可以很好地处理变长的输入. 通过 LSTM 来生成句子的表示跟传统的基于词袋模型[32] 的方法是完全不同的. 在词袋模型的表示方法中, 每一个句子按照词是否出现与词的频率将句子转换成一个词向量, 每一个维度代表词的 TF-IDF 值等权重. 词袋模型并没有考虑词与词之间的依赖关系等. LSTM 生成的句子表示捕获了句子的依赖结构和词与词之间的相似度, 因此可以更好地表示句子的语义. 因此, 在本文中, 我们采用长短时记忆模型来将输入代码映射成语义向量, 这个向量代表了代码全局的信息.

在每一步, 一个新的词嵌入向量被输入到长短时记忆模型中. 长短时记忆模型是深度学习模型的一种. 它的主要思想是顺序地处理句子中的每一个词, 在读取到最后一个词时, LSTM 的输出将前文压缩成一个稠密的向量, 这个向量可被认为是位于

一个低维的语义向量空间中. 最终的目标为: 语义相似句子的向量在该语义向量空间中比较近, 而语义不同的句子的向量在语义空间当中的距离比较远.

LSTM 扩展自循环神经网络 (Recurrent neural network, RNN)[33], 给定一个输入 x_t 与之前的输出 h_{t-1}, 循环神经网络计算出下一个输出 h_t 为

$$l_t = W_h x_t \tag{1}$$

$$h_t = \sigma(W l_t + W_r h_{t-1} + b) \tag{2}$$

其中, W 与 W_r 是模型的参数, b 是偏差, l_t 是隐含状态. 尽管循环神经网络可以将句子转换成相应的向量形式并且在文献 [34] 中被证明是图灵完备的, 但是循环神经网络由于梯度消失的现象变得在实际中难以优化学习. 当输入序列较长时, 梯度在反向传播的时候很快变得接近于零导致几乎不能学习.

LSTM 通过控制信息的流入流出来解决梯度消失的现象. 图 3 为 LSTM 单元的示意图. LSTM 单元有三种类型的控制门: 输入门、遗忘门 (Forget gate) 和输出门. LSTM 在输入较长时比 RNN 更好地捕获输入的依赖关系. 类似于 RNN, LSTM 顺序地更新隐含状态. LSTM 有记忆状态 c_t, 输入状态 i_t 和遗忘状态 f_t. 给定输入向量 x_t, 上一单元的输出 h_{t-1}, 上一单元状态 c_{t-1}, 单元的输出 h_t 计算为

$$i_t = \sigma(W_{xi} x_t + W_{hi} h_{t-1} + W_{ci} c_{t-1} + b_i) \tag{3}$$

$$f_t = \sigma(W_{xf} x_t + W_{hf} h_{t-1} + W_{cf} c_{t-1} + b_f) \tag{4}$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc} x_t + W_{hc} h_{t-1} + b_c) \tag{5}$$

$$o_t = \sigma(W_{xo} x_t + W_{ho} h_{t-1} + W_{co} c_t + b_o) \tag{6}$$

$$h_t = o_t \tanh(c_t) \tag{7}$$

在上面的式子中, 一系列的 W 矩阵和偏差 b_i, b_f, b_c, b_o 为要学习的参数, σ 表示 Sigmoid 函数. 在输入最后一个词后, 最后的输出作为代码的语义向量, 这个向量可以捕获代码的上下文语义信息, 并作为代码的表示输入到之后的处理中.

得到智能合约代码的向量表示之后, 我们将向量表示与智能合约交易的信息组合起来输入到前馈神经网络中, 最终的输出为该智能合约代码可能的类标签及其对应概率. 在下一节我们介绍如何生成智能合约相关交易的信息.

4.3　交易信息提取

如图 1 所示, Ethereum 区块链上保存了关于智能合约账号的所有信息[35−36]. 每一个智能合约账号对应着一个 160 位的地址, 通过地址可以唯一确定一个智能合约.

在提取了智能合约代码的语义向量后, 我们将利用智能合约相关交易记录及交易所产生的账号信息来进一步对智能合约建模.

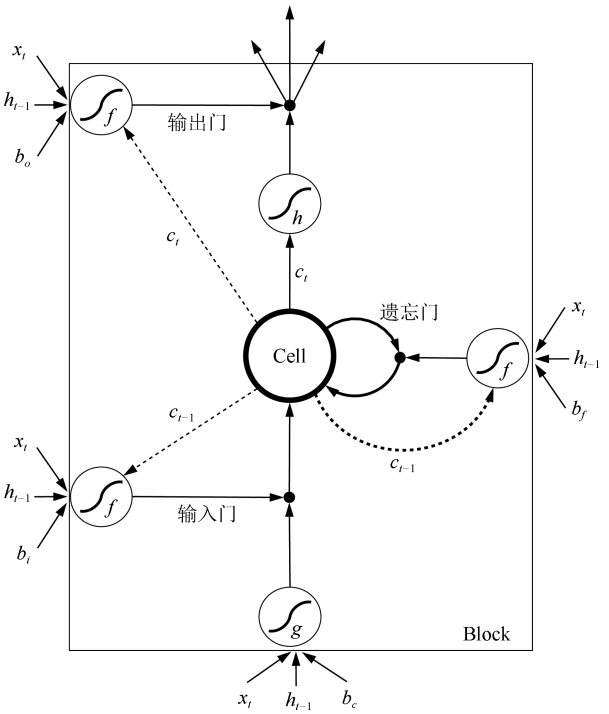


图 3　LSTM 单元

Fig. 3　LSTM unit

我们选用的智能合约账号信息特征包括:

1) 智能合约和创建者账号余额: Ethereum 账号余额以 Wei 为单位.

2) 智能合约的 Nonce: Nonce 是一个计数器, 记录着该智能合约是创建者创建的第几个智能合约.

3) 创建者的 Nonce: 创建者的 Nonce 记录着创建者关联交易的数量

4) 所在区块的块号: 智能合约账号创建时所在区块的块号或者区块高度.

5) 智能合约和创建者账号的 PageRank 值: 我们构建了一个交易活动图, 图的节点是用户账号和智能合约账号, 边用区块链区块中的交易记录来构建, 比如账号 A 转账给账号 B, 则加一条从 A 到 B 的边. 通过计算图上各个节点的 PageRank 值来衡量节点在 Ethereum 区块链上的重要性和交易的活跃程度.

一般用户账号可以对智能合约账号发起两种关联交易, 第一种是创建一个新的智能合约账号, 第二种是调用智能合约定义的函数. 我们采用的关联交易信息特征包括:

1) 交易描述性统计: 关联交易的总数量、关联交易金额的平均值和方差、关联交易交易费的平均值和方差、关联交易交易费的平均费率 (Gas price).

2) 关联交易发起账号描述性统计: 账号平均余额和方差、账号平均发起交易数目和方差.

3) 智能合约有无发起交易: Ethereum 允许智能合约发起新的交易, 比如一个彩票智能合约会发起交易转账给获奖者. 如果有这样的交易, 计算交易的总数量、平均数额与方差; 否则, 为零.

4) 关联交易时间戳: 关联交易发起时间戳差的平均值与方差、最后一次交易距离现在的时间、最早一次交易距离现在的时间.

5) 关联交易调用的智能合约函数: 调用智能合约函数的数量、调用的数量除以智能合约函数的总数、调用频率.

在得到智能合约的这些特征向量后, 我们将把这个向量与语义向量组合起来, 预测最终的类标签概率.

4.4 类标签预测

在得到智能合约的代码语义向量 V 与交易信息特征向量 T 后, 我们连接两个向量 (V,T), 组合后的向量可以完整的代表智能合约的相关信息.

组合后的向量 (V,T) 作为前馈神经网络的输入层, 前馈神经网络有一个隐藏层, 最后一层为输出层, 输出类标签的概率, 因此输出层节点的个数等于 $|\{C_1, C_2, \cdots\}|$, 即类标签的个数. 下面详细介绍每一层.

隐藏层: 在给定输入 (V,T) 下, 隐藏层进行了如下转换:

$$\alpha(W_h(\dot{V},T)+b_h) \tag{8}$$

其中, W_h 为隐藏层的参数矩阵, α 为激活函数, b_h 为偏差. 我们在输入层和输出层之间加一个隐藏层的目的是允许我们的模型获得非线性的分类能力.

为了让神经网络学习到一个非线性的决策边界, 非线性的 α 选择有 Sigmoid, tanh, ReLU (定义为 $\max(0,x)$). 激活函数的选择会影响模型的收敛速度和最优解的选择. 在文献 [37] 中证明了 ReLU 的性能显著好于另外两种. Sigmoid 和 tanh 在被激活后梯度接近于 0, 因此导致模型学习速度较慢. 因此在本文中, 我们选择 ReLU 作为激活函数.

输出层: 在得到隐藏层的输出 x 后, 为了最后的输出是一个概率分布, 我们将输出传到一个 Softmax 层中. Softmax 层计算得到一个关于类标签的概率分布:

$$p(y=C_j|x)=\frac{\mathrm{e}^{x^{\mathrm{T}}\theta_j}}{\sum_k \mathrm{e}^{x^{\mathrm{T}}\theta_k}} \tag{9}$$

θ_k 是第 k 类的权重向量, 为了得到类标签 C_j 的概率, 我们用 $\sum_k \mathrm{e}^{x^{\mathrm{T}}\theta_k}$ 来归一化.

损失函数: 在得到输出层的概率输出后, 为了学习模型的参数, 我们用标准的交叉熵函数来作为损失函数:

$$L(p,q)=-\sum_k p(y=C_k|x)\log q(y=C_k) \tag{10}$$

概率分布 p 即输出层的输出, 概率分布 q 由训练数据 $\{D_i, y_i\}$ 的标签决定, 因此对一个数据点 (D_i, y_i), 在 y_i 上为 1, 在其他坐标为 0.

正则化: 虽然神经网络有强大的性能来学习复杂的决策函数, 神经网络在中小型的数据集上很容易过拟合. 鉴于 Ethereum 上智能合约的总数量仍然较少, 我们用参数的 l_2 模来缓解过拟合的现象.

我们同时用 Dropout[38] 来处理过拟合的现象. Dropout 是一种有效地处理神经网络过拟合的方式. 通过舍弃一些隐藏层节点的输出, Dropout 可以让一些参数在训练时维持不变. Dropout 可以认为是近似地对一系列模型取平均[39]. 通过设置 Dropout 的参数, Dropout 可以有效地控制参数的变化.

训练过程: 模型的参数用随机梯度下降来获得梯度, 用反向传播来训练. 为了提高模型的训练速度, 有许多随机梯度下降更新规则被提出: Adagrad[40]、Adadelta[41]. Adadelta 利用误差梯度历史记录和权重更新记录来更新梯度, 避免人为设置不合适的学习速率, 因此我们选择 Adadelta 来学习模型的参数.

训练完成后, 我们得到了一个可以预测智能合约类标签的模型. 我们将通过一系列实验来验证它的性能.

5 实验结果与分析

5.1 数据集采集与标记

为了验证模型的性能, 我们需要构建一个智能合约的数据集. 我们采用一种弱标签的方式来标记智能合约的数据集.

我们提取了 Ethereum 前 1 480 000 个区块的所有信息, 区块链的信息由 EtherChain[4] 上抓取获得. 我们首先从区块中提取出包含的智能合约, 然后标记智能合约的类别.

通过账号的类别可以直接从区块链中提取智能合约, 21 214 个智能合约被收集. 这些智能合约的余额加起来超过三百万 Ether, 目前价值超过三千万美元. 智能合约的余额有显著的差异, 大体服从 Zipf 法则. 只有大约 10 % 的账号中有超过 1 Ether, 账号的平均 Ether 价值是 4 612 美元. 智能合约的复杂程度也有显著差异, 智能合约指令的数量从 18 条到 23 869 条不等, 平均有 2 565 条. 我们将交易不活跃的和有编译等问题的智能合约去除之后, 剩余

[4] https://etherchain.org/

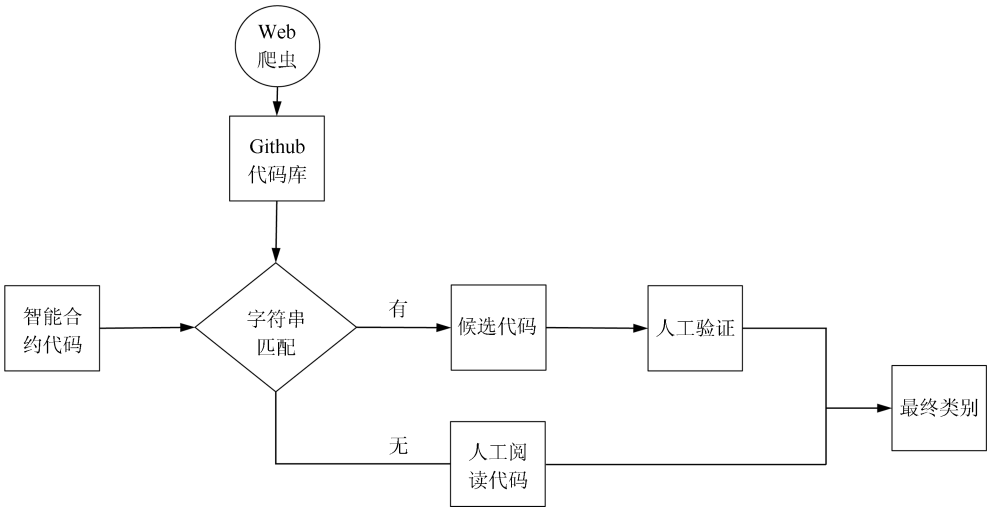


图 4　标记流程

Fig. 4　Mark process

11 176 个智能合约.

下一步, 我们需要为智能合约添加类标签. 人工地阅读理解代码并添加类标签非常地消耗时间. 为了提高智能合约的标记速度, 我们设计了如图 4 所示的类标签标记流程. 我们下载了代码托管平台 Github[5] 上关于 Ethereum 智能合约的公共代码库; 然后, 通过代码字符串相似度匹配, 找出相似度最高的代码, 通过人工来分析最终是否匹配; 最后, 通过阅读 README 等信息快速地标记智能合约的类标签. 如果没有找到相关的公共代码库, 我们通过搜索智能合约的地址或代码内容到 Google, 如果有相关的信息, 相关的网页内容可以被用来快速地标记智能合约的类标签. 通过上述两种方式, 我们标记了超过一万个智能合约的类标签. 余下的智能合约通过阅读代码完成了类标签标记.

5.2　智能合约类别与实验设定

通过分析智能合约的相关应用, 我们将智能合约的用途分为: 金融类 (保险、融资、投资等)、游戏类、彩票类、Ethereum 工具类 (如 Ethereum 区块链的可视化程序)、信息管理类 (用户身份管理等)、货币类、娱乐类 (音乐、视频、社交等)、物联网类与其他. 各个类别的数量如图 5 所示. 从图 5 中我们可以看出金融类的应用最多, 其次是货币类和 Ethereum 工具类. 因此类标签的数量并不均衡.

本文所有实验均由一台 Intel Xeon E5 四核的电脑完成, CPU 主频为 3.7 GHz, 内存为 8.00 GB. 程序使用 Java JDK 1.8 编写完成. 鉴于 Ethereum 区块链数据集规模在单机可处理的范围, 本文的算法无需使用集群训练与验证, 因此可以方便地移植到实际应用中.

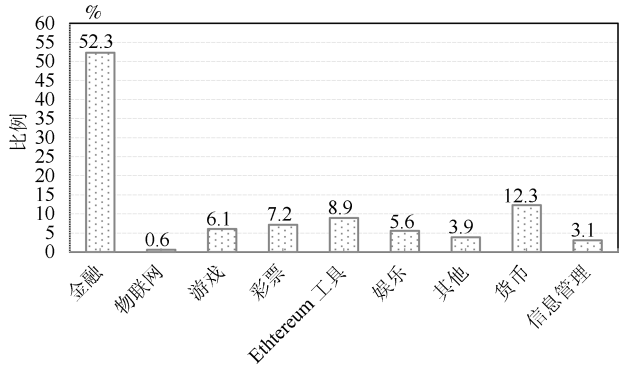


图 5　类别统计

Fig. 5　Category statistics

5.3　算法有效性

算法的目标是有效地将智能合约的代码分类. 为了验证算法的性能, 我们将算法与其他几种常用的分类算法比较:

1) 朴素贝叶斯: 为了与词嵌入模型对比, 我们直接用词袋模型将代码转换成向量, 然后连接智能合约的交易信息作为朴素贝叶斯的输入, 朴素贝叶斯计算出每一个类别的概率, 我们取概率最大的一个作为智能合约的预测.

2) 支持向量机: 支持向量机是一种常用的做文本分类的算法, 我们用支持非线性分类的高斯核的支持向量机作为分类器. 支持向量机使用的特征与朴素贝叶斯分类器一致. 为了让支持向量机支持多类分类, 我们采用 One VS All 的策略来进行多类分类.

我们采用了 Precision、Recall、Accuracy 和 F1

[5] http://github.com

score 四种评估模型分类效果的常用评估方法. 表 1～3 是三种分类器在 9 种类别上的性能. 为了评估交易信息对分类的影响, 我们同时评估了有或者没有交易信息情况下的分类结果.

如表 1～3 所示, 我们的模型有效性明显好于朴素贝叶斯模型与支持向量机模型, 以金融类为例, 在 F1 score, 我们的算法有交易信息和无交易信息的情况下分别取得了 94.3 % 和 86.9 % 的分类准确率. 在有交易信息的情况下, 相比于朴素贝叶斯算法和支持向量机分别取得了 7 % 和 6 % 的提升. 在无交

表 1 神经网络分类效果

Table 1 Neural network classification effect

类别	有交易信息				无交易信息			
	Precision	Recall	Accuracy	F1 score	Precision	Recall	Accuracy	F1 score
金融类	0.943	0.945	0.942	0.943	0.872	0.868	0.882	0.869
游戏类	0.924	0.897	0.924	0.910	0.895	0.874	0.886	0.884
彩票类	0.882	0.891	0.906	0.886	0.835	0.852	0.875	0.843
Ethereum 工具类	0.914	0.921	0.929	0.917	0.854	0.871	0.882	0.862
信息管理类	0.862	0.842	0.883	0.852	0.805	0.813	0.829	0.809
货币类	0.914	0.882	0.917	0.898	0.821	0.809	0.834	0.814
娱乐类	0.873	0.889	0.893	0.881	0.783	0.763	0.792	0.773
物联网类	0.861	0.845	0.882	0.853	0.796	0.771	0.809	0.783
其他	0.832	0.814	0.845	0.823	0.753	0.757	0.791	0.754

表 2 朴素贝叶斯分类效果

Table 2 Naive Bayesian classification effect

类别	有交易信息				无交易信息			
	Precision	Recall	Accuracy	F1 score	Precision	Recall	Accuracy	F1 score
金融类	0.862	0.893	0.861	0.877	0.861	0.815	0.862	0.837
游戏类	0.866	0.879	0.883	0.872	0.815	0.826	0.837	0.820
彩票类	0.821	0.817	0.846	0.819	0.796	0.805	0.822	0.800
Ethereum 工具类	0.884	0.854	0.896	0.868	0.825	0.847	0.861	0.835
信息管理类	0.829	0.859	0.860	0.852	0.757	0.771	0.796	0.764
货币类	0.876	0.853	0.896	0.864	0.760	0.765	0.774	0.762
娱乐类	0.845	0.864	0.872	0.854	0.716	0.725	0.735	0.720
物联网类	0.826	0.843	0.862	0.834	0.746	0.741	0.759	0.743
其他	0.784	0.819	0.825	0.801	0.745	0.737	0.763	0.740

表 3 支持向量机分类效果

Table 3 Support vector machine classification effect

类别	有交易信息				无交易信息			
	Precision	Recall	Accuracy	F1 score	Precision	Recall	Accuracy	F1 score
金融类	0.875	0.897	0.906	0.885	0.815	0.831	0.842	0.822
游戏类	0.883	0.835	0.876	0.858	0.845	0.821	0.856	0.832
彩票类	0.879	0.846	0.887	0.862	0.855	0.793	0.814	0.822
Ethereum 工具类	0.861	0.865	0.891	0.862	0.829	0.827	0.836	0.827
信息管理类	0.804	0.863	0.877	0.832	0.764	0.786	0.789	0.774
货币类	0.872	0.862	0.889	0.866	0.787	0.792	0.803	0.789
娱乐类	0.863	0.859	0.873	0.860	0.708	0.714	0.726	0.710
物联网类	0.829	0.845	0.867	0.836	0.756	0.758	0.763	0.756
其他	0.804	0.821	0.856	0.812	0.731	0.727	0.734	0.728

易信息的情况下, 取得了 3% 和 4% 的提升. 朴素贝叶斯跟支持向量机利用词袋模型来建立文本特征, 词袋模型可以有效地捕获词性上的特征, 但是, 词袋模型忽略了代码中的词顺序和依赖关系. 我们的系统利用词嵌入模型来捕获词语义信息. 同时为了更好的捕获代码的全局特征, 我们利用一个单层的 LSTM 来生成代码的全局语义向量. 我们顺序地输入词的嵌入向量, 因此 LSTM 可以更好地捕获代码中的顺序和依赖关系. 并且词嵌入模型不需要人工的选择特征, 因此可以被更好地应用在其他领域中. 我们注意到, 支持向量机模型的效果略微好于朴素贝叶斯算法, 原因在于, 朴素贝叶斯算法是一种线性模型, 并不能很好地用于非线性的关系当中, 支持向量机利用高斯核函数较好地解决了这个问题. 神经网络和支持向量机都可以支持分线性的分类边界, 但是多类别的分类问题中, 支持向量机需要用 One Vs All 等策略来间接的将多元问题转换成二元问题. 神经网络可以按类别数目直接输出类别的概率分布. 因此, 我们的系统相比于支持向量机有更好的可解释性.

另一方面, 在不加入交易信息特征向量的情况下, 三种方法的准确率都有显著的下降. 在没有交易信息的情况, 模型只能从代码本身来判断代码的类别, 交易信息可以从外部对它的调用中提供关于智能合约的行为信息. 因此, 我们断言, 通过智能合约关联的交易, 模型可以获得关于智能合约更多的信息. 通过 t 检验, 我们的实验结果有统计显著性.

在 9 种不同类别上, 模型在金融类、游戏类和货币类上的分类性能最好. 在其他类上的表现较差, 原因可能是其他类包含的隐含种类较多, 导致模型无法很好地学习到分类边界. 鉴于此, 我们将在未来工作中更细粒度的分析类标签信息, 比如用树或者图来表示类标签的关系.

通过以上的实验, 我们验证了模型的有效性. 我们的算法可以被应用在预测智能合约代码的类别中并取得明显优于其他对比方法的准确率.

5.4 系统的应用

为了验证系统的可行性并且更好地应用本系统, 我们利用本文提出的算法实现了一个智能合约的浏览器与智能合约的检索系统.

我们按照实验中提出的 9 种类别, 按类别组织智能合约, 用户可以按类别去浏览智能合约, 同时我们连接智能合约到与之相关的代码库. 当有新的代码在 Ethereum 区块链上被生成时, 我们调用神经网络去自动分类它的类别, 并加到相关的代码类别中去. 用户可以通过本智能合约浏览器更好地了解智能合约目前的发展情况与应用环境.

另一方面, 我们实现了一个智能合约的检索系统. 首先, 我们利用 LSTM 的输出作为每一个智能合约的向量表示. 用户可以用关键词去搜索相关的智能合约. 输入的关键词首先通过 LSTM 转换成一个语义向量. 然后, 我们用这个语义向量与代码库中的向量计算余弦相似性. 最后按照余弦性由高到低排序. 用户可以方便地检索到自己需要的代码示例.

6 结束语

本文以 Ethereum 平台上的智能合约为例, 研究了基于词嵌入模型与交易信息的智能合约代码分类系统. 通过用词嵌入模型对智能合约代码的语义信息建模, 并且用智能合约的交易信息来更深入地理解智能合约的逻辑行为, 最后用神经网络来组合两方面的信息输出最终的类标签概率分布. 实验结果证实了系统的可行性和有效性. 浏览器与智能合约检索系统验证了本系统广泛的应用前景. 为了进一步改进系统的性能, 我们将在接下来的研究中考虑使用层次化的类标签结构, 同时考虑更复杂更完整的类标签信息的处理.

References

1 Nakamoto S. Bitcoin: a peer-to-peer electronic cash system, http://www.bitcoin.org, September 7, 2017

2 Castro M, Liskov B. Practical byzantine fault tolerance. In: Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI), USENIX Association, 1999, **99**: 173−186

3 Pang G S, Jin H D, Jiang S Y. Cenknn: a scalable and effective text classifier. *Data Mining and Knowledge Discovery*, 2015, **29**(3): 593−625

4 Tang B, He H B, Baggenstoss P M, Kay S. A Bayesian classification approach using class-specific features for text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 2016, **28**(6): 1602−1606

5 Wahiba B A, El Fadhl Ahmed B. New fuzzy decision tree model for text classification. In: Proceedings of the 1st International Conference on Advanced Intelligent System and Informatics (AISI2015). Switzerland: Springer, 2016. 309−320

6 Mikolov T, Sutskever I, Chen K, Corrado G S, Dean J. Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems 26. Lake Tahoe, Nevada, United States: Curran Associates Inc., 2013. 3111−3119

7 Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv: 1409.0473, 2014.

8 Liu B. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 2012, **5**(1): 1−167

9 Fleder M, Kester M S, Pillai S. Bitcoin transaction graph analysis. arXiv preprint arXiv: 1502.01657, 2015.

10 Ron D, Shamir A. Quantitative analysis of the full bitcoin transaction graph. In: Proceedings of the 17th International Conference on Financial Cryptography and Data Security. Okinawa, Japan: Springer, 2013. 6−24

11 Shah D, Zhang K. Bayesian regression and bitcoin. In: Proceedings of the 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton). Monticello, USA: IEEE, 2014. 409−414

12 Luu L, Chu D H, Olickel H, Saxena P, Hobor A. Making smart contracts smarter. Cryptology ePrint Archive, Report 2016/633 [Online], available: http://eprint.iacr.org/2016/633, August 16, 2016.

13 Moore T, Christin N. Beware the middleman: empirical analysis of bitcoin-exchange risk. In: Proceedings of the 17th International Conference on Financial Cryptography and Data Security. Okinawa, Japan: Springer, 2013. 25−33

14 Omohundro S. Cryptocurrencies, smart contracts, and artificial intelligence. *AI Matters*, 2014, **1**(2): 19−21

15 Di Battista G, Di Donato V, Patrignani M, Pizzonia M, Roselli V, Tamassia R. Bitconeview: visualization of flows in the bitcoin transaction graph. In: Proceedings of the 2015 IEEE Symposium on Visualization for Cyber Security (VizSec). Chicago, USA: IEEE, 2015. 1−8

16 Sebastiani F. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, 2002, **34**(1): 1−47

17 Rocchio J J. Relevance feedback in information retrieval. *The SMART Retrieval System*. Englewood Cliffs, N.J.: Prentice Hall, Inc., 1971.

18 Rao Y H, Li Q, Mao X D, Liu W Y. Sentiment topic models for social emotion mining. *Information Sciences*, 2014, **266**: 90−100

19 Rao Y H, Xie H R, Li J, Jin F M, Wang F L, Li Q. Social emotion classification of short text via topic-level maximum entropy model. *Information & Management*, 2016, **53**(8): 978−986

20 Salton G, Wong A, Yang C S. A vector space model for automatic indexing. *Communications of the ACM*, 1975, **18**(11): 613−620

21 Liu M Y, Yang J G. An improvement of TFIDF weighting in text categorization. In: Proceedings of the 2012 International Conference on Computer Technology and Science. Singapore: IACSIT Press, 2012. 44−47

22 Li C H, Park S C. Combination of modified BPNN algorithms and an efficient feature selection method for text categorization. *Information Processing and Management*, 2009, **45**(3): 329−340

23 Hinton G E, Salakhutdinov R R. Reducing the dimensionality of data with neural networks. *Science*, 2006, **313**(5786): 504−507

24 Chen Z H, Ni C W, Murphey Y L. Neural network approaches for text document categorization. In: Proceedings of the 2006 IEEE International Joint Conference on Neural Network. Vancouver, Canada: IEEE, 2006. 1054−1060

25 Li C H, Song W, Park S C. An automatically constructed thesaurus for neural network based document categorization. *Expert Systems with Applications*, 2009, **36**(8): 10969−10975

26 Turian J, Ratinov L, Bengio Y. Word representations: a simple and general method for semi-supervised learning. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010. 384−394

27 Pennington J, Socher R, Manning C D. GloVe: global vectors for word representation. In: Proceedings of the Empiricial Methods in Natural Language Processing, 2014, **12**: 1532−1543

28 Le Q V, Mikolov T. Distributed representations of sentences and documents. In: Proceedings of the 31th International Conference on Machine Learning. Beijing, China, 2014. 1188−1196

29 Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Computation*, 1997, **9**(8): 1735−1780

30 Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems 27. Montreal, Quebec, Canada: MIT Press, 2014.

31 Tim R, Grefenstette E, Hermann K M, Tomáš K, Blunsom P. Reasoning about entailment with neural attention. arXiv preprint arXiv: 1509.06664, 2015.

32 Huang P S, He X D, Gao J F, Deng L, Acero A, Heck L. Learning deep structured semantic models for web search using clickthrough data. In: Proceedings of the 22nd ACM International Conference on Information & Knowledge Management. New York, NY, USA: ACM, 2013. 2333−2338

33 Mikolov T, Karafiát M, Burget L, Černocký J, Khudanpur S. Recurrent neural network based language model. In: INTERSPEECH 2010, Conference of the International Speech Communication Association. Makuhari, Chiba, Japan: ISCA, 2010. 1045−1048

34 Siegelmann H T, Sontag E D. On the computational power of neural nets. In: Proceedings of the 5th Annual Workshop on Computational Learning Theory. New York, NY, USA: ACM, 1992. 440−449

35 Buterin V. Ethereum white paper [online], available: https://github.com/ethereum/wiki/wiki/White-Paper, September 7, 2017

36 Wood G. Ethereum: a secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper, 2014.

37 Maas A L, Hannun A Y, Ng A Y. Rectifier nonlinearities improve neural network acoustic models. In: Proceeding of the 2013 ICML Workshop on Deep Learning for Audio, Speech, and Language Processing. Atlanta, Georgia, 2013.

38 Srivastava N, Hinton G E, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 2014, **15**(1): 1929−1958

39 Goodfellow I J, Warde-Farley D, Mirza M, Courville A C, Bengio Y. Maxout networks. *ICML*, 2013, **28**(3): 1319−1327

40 Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 2011, **12**: 2121−2159

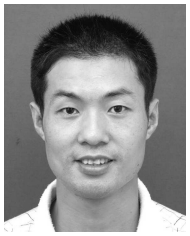
41 Zeiler M D. Adadelta: an adaptive learning rate method. arXiv preprint arXiv: 1212.5701, 2012.

黄步添 浙江大学计算机科学与技术学院博士研究生. 主要研究方向为虚拟化, 云计算, 区块链. 本文通信作者.
E-mail: butine@zju.edu.cn
(**HUANG Bu-Tian** Ph. D. candidate at the College of Computer Science and Technology, Zhejiang University. His research interest covers virtualization, cloud computing, and blockchain. Corresponding author of this paper.)

刘 琦 新加坡国立大学计算机学院硕士研究生. 主要研究方向为数据挖掘, 区块链.
E-mail: leuchine@gmail.com
(**LIU Qi** Master student at the College of Computer Science, National University of Singapore, Singapore. His research interest covers data mining and blockchain.)

何钦铭 浙江大学计算机科学与技术学院教授. 主要研究方向为数据挖掘, 虚拟化, 区块链. E-mail: hqm@zju.edu.cn
(**HE Qin-Ming** Professor at the College of Computer Science and Technology, Zhejiang University. His research interest covers data mining, virtualization, and blockchain.)

刘振广 新加坡国立大学计算机学院博士后. 主要研究方向为数据挖掘, 区块链. E-mail: zhenguangliu@zju.edu.cn
(**LIU Zhen-Guang** Postdoctor at the College of Computer Science, National University of Singapore, Singapore. His research interest covers data mining and blockchain.)



陈建海 浙江大学计算机科学与技术学院讲师. 主要研究方向为虚拟化, 云计算, 区块链. E-mail: chenjh919@zju.edu.cn
(**CHEN Jian-Hai** Lecturer at the College of Computer Science and Technology, Zhejiang University. His research interest covers virtualization, cloud computing, and blockchain.)