Computer Science and Application 计算机科学与应用, 2020, 10(4), 741-748
Published Online April 2020 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2020.104077

Hans汉斯

The Application of LSH Technology in Similar Question Detection

Rui Chen[1], Song Wang[1], Ying Mei[1], Yunyuan Yang[2]

[1]School of Economics and Management, Chuxiong Normal University, Chuxiong Yunnan
[2]School of Geographical Science and Tourism Management, Chuxiong Normal University, Chuxiong Yunnan
Email: 308146626@qq.com

Received: Apr. 2nd, 2020; accepted: Apr. 17th, 2020; published: Apr. 24th, 2020

Abstract

The repetition rate of test questions is one of the important indexes to evaluate the quality of test questions and test papers. In order to quickly find out similar questions in the test bank, this paper mainly studies the detection methods of similar questions based on K-shingles, Jaccard similarity, MinHash and LSH technology. First of all, the main content of the question is segmented into Chinese words, then converted into K-shingle set after proper processing, and the signature is calculated by MinHash. Finally, LSH technology is used to quickly find out the candidate pairs of similar questions and calculate the corresponding Jaccard similarity. If the similarity is greater than the given threshold, similar questions are found. Experiments prove to be practicable and effective.

Keywords

Examination Checking, LSH Algorithm, Jaccard Similarity, K-Shingle

基于LSH技术的试题相似度检测方法

陈　瑞[1]，王　松[1]，梅　莹[1]，杨云源[2]

[1]楚雄师范学院经济与管理学院，云南　楚雄
[2]楚雄师范学院地理科学与旅游管理学院，云南　楚雄
Email: 308146626@qq.com

收稿日期：2020年4月2日；录用日期：2020年4月17日；发布日期：2020年4月24日

摘　要

试题内容重复率是评价试题库及试卷质量的重要指标之一，为了快速找出题库中的相似试题，本文主要

文章引用: 陈瑞, 王松, 梅莹, 杨云源. 基于 LSH 技术的试题相似度检测方法[J]. 计算机科学与应用, 2020, 10(4): 741-748. DOI: 10.12677/csa.2020.104077

研究了基于K-shingles的Jaccard相似度、MinHash和LSH技术应用于相似试题的检测方法。此方法首先将题干内容进行中文分词，进行适当处理后转换成K-shingle集，通过MinHash计算出签名，最后使用LSH技术快速地找出候选相似试题对并计算出相应的Jaccard相似度，若该相似度大于给定的阈值，则发现相似试题。该方法通过在题库系统中的使用，充分验证了该方法的可行性，达到了很好的效果。

关键词

试题查重，LSH算法，Jaccard相似度，K-shingle

1. 引言

试题内容重复是影响试题库及试卷质量的重要因素之一，所以对相似试题的检测和去重就尤为重要。目前相似试题的发现方法主要使用自然语言处理(NLP)中的文本相似性检测方法。这些方法大致可分为以下几类：

- 单词共现方法(Word co-occurrence)。
- 基于词汇数据库(lexical database)的相似度方法。
- 基于词向量(word vectors)的递归神经网络和深度神经网络方法。

单词共现方法通常用于识别文本数据中的重要部分[1]。该方法具有明显的缺点，例如：

- 它忽略了句子的词序。
- 它不考虑单词在句子上下文中的含义。

但是它具有以下优点：

- 可匹配任何文本。
- 能成功地从中提取关键字。

词汇数据库方法，使用具有在树状结构中编译的词、含义和与其他词关系的预定义词层次来计算相似度[2]。在比较两个词时，它考虑了词之间的路径距离以及包含者在层次中的深度。包含者指的是与被比较的两个词有关的相对根节点。它还使用单词语料库来计算单词的“信息内容”，这会影响最终的相似度。该方法具有以下局限性：

- 在计算相似度时不考虑单词的适当含义，而是选择最佳匹配对，即使单词在两个不同的句子中的含义完全不同。
- 来自一个语料库的词的信息内容与另一个语料库中的词的信息内容不同。

近年来，基于神经网络的模型在语义相似度有了很大的改进[3] [4] [5] [6] [7]。其中乐雨泉等[8]提出了一个称为 ACV-tree 句子建模方法。这个方法可以看成一种通用的句子建模框架，此框架试图将句法知识(syntactic information)、语义信息(semantic information)、注意力权重机制(attention weight mechanism)合并到一个统一的结构中来吸收它们的优点。解决了自然语言处理中句子建模的问题。梁圣[9]提出了循环神经网络用于计算试题相似度的方法，对比了不同词向量维度以及 LSTM 隐含层神经单元个数对相似度计算结果的影响。田星等[10]提出了一种基于词向量的 Jaccard 句子相似度算法，该方法首先通过训练将每个词语映射为语义层面的高维向量，然后计算各个词向量之间的相似度，高于阈值 α 的作为共现部分，

从而计算句子的相似度。陈勤等[11]提出了一种基于上下文对齐的 RNN (CA-RNN)模型，该模型将对齐后的词的上下文信息整合到一个句子对中，用于内部 hidden state 的生成。虽然神经网络方法在语义的相似性上取得了很大的改善，但神经网络方法却需要成千上万个标记样本作为训练数据，而训练往往需要花费大量的时间。

总体而言，上述所有方法在试题相似性检测上往往不太适合。这些方法过于追求于语义上的精准相似，代价高昂。其忽略了在多个样本中，两两相似比较的时间复杂度。本文的研究方法，即不需要额外的语料数据库，也不需要大量的训练样本，更无需借助除了 Office 之外的其它功能即可快速地实现相似试题的检测。

该方法基于 K-shingle，K-shingle 是连续 K 个词的唯一序列，通过 K-shingles 将试题内容转换成一组位向量 0 和 1 来表示，其中，0 表示该试题中不存在此 shingle，1 表示该试题存在此 shingle。近似相同的两个试题应该有许多相同的 shingle。两个试题内容是否相似可以通过使用 Jaccard 相似度来计算得到。Jaccard 相似度在数据集较小时，相似对的比较和相似度计算都表现良好。然而，在数据集较大时，时间复杂度将几何上升，此时可以通过 MinHash 转换的签名矩阵和位置敏感散列(LSH)技术来解决这个问题，从而在更短的时间内快速逼近真正的 Jaccard 相似度。

2. 研究内容

2.1. Jaccard 相似度

两个集合 M 和 N 的 Jaccard 相似度是 $|M \cap N|/|M \cup N|$，即两个集合交集的数目与两个集合并集数目的比例(Rajaraman and Ullman, 2011) [12]。

$$JSim(M,N)=\frac{|M \cap N|}{|M \cup N|}$$

Jaccard 相似度具有如下性质：当且仅当两个集合相等时 $M=N$，$JSim(M,N)=1$；当且仅当 $M \neq N$，$JSim(M,N)=0$； $JSim$ 是对称的。基于 Jaccard 相似度的文档相似度在很多方面均有应用，如：文档抄袭、镜像页面、同源新闻稿等。如下面两个集合的相似度为 $JSim(M,N)=1$。

若有两个集合 $A=\{a,b,c,d,e\}, B=\{c,d,e,f,g,h\}$，则 $JSim(A,B)=3/8$。

2.2. 中文分词

在中文中，词语之间没有分隔，且边界模糊。因此进行中文自然语言处理通常是先将汉语文本中的字符串切分成合理的词语序列，然后再在此基础上进行其它分析处理。中文分词是中文信息处理的一个基础环节，已被广泛应用于中文文本处理、信息提取、文本挖掘等应用中。例如：T1 = {“关系数据库理论包括函数依赖和_____”}，经过分词之后为 Ts1 = {“关系”，“数据库”，“理论”，“包括”，“函数”，“依赖”，“和”}。

2.3. 文档的 K-shingles

为了识别字面上相似的文档，将文档表示成集合的最有效的方法是构建文档中的短字符串集合(即 shingle)。如果文档采用这样的集合表示，那么有相同句子甚至短语的文档之间将会拥有很多公共的 shingle。其中 K 表示子串的长度。为了通过 K-shingle 表示一个文档，需要提前对文档进行处理。如删除首尾空格、删除停用词等。然后再把文本拆分成分词。例如经过分词之后的 $T1$ = {“关系”，“数据库”，“理论”，“包括”，“函数”，“依赖”}，如果 K = 2，则通过 K-shingle，$Ts1$ = {“关系数据库”，“数据库理论”，“理论包括”，“包括函数”，“函数依赖”}，如果还具有下列两个文本 $T2$ = {“数

据库的理论包括函数依赖和_____”}、$T3$ = {“关系数据库理论包括_____和规范化”}，则 3 个文本的 shingle 元素矩阵如表 1 所示。

Table 1. Shingle element matrix of text
表 1. 文本的 shingle 元素矩阵

shingles	$T1$	$T2$	$T3$
关系数据库	1	0	1
数据库理论	1	1	1
理论包括	1	1	1
包括函数	1	1	0
函数依赖	1	1	0
包括规范化	0	0	1

表 1 中的文本对相似度如下： $JSim(T1,T2)=4/5$， $JSim(T1,T3)=3/6$， $JSim(T2,T3)=2/6$。

2.4. Shingle 的哈希

由于所有 shingle 集合的存储需要占用大量的空间。可以哈希 shingle 进行散列，使用散列之后的桶号来代替 shingle 串。我们使用以下形式的散列函数[13]：

$$h(x)=(ax+b)mod\ p$$

其中： x 为原始矩阵中的 shingle 串，a 和 b 是从(0, Shingle 全集大小)之间的一个随机数，p 是比 Shingle 全集稍大的数。

假定表 1 中的 shingle 串被哈希到 0~5 的桶中，见表 2。

Table 2. Characteristic matrix M
表 2. 特征矩阵 M

shingles	$T1$	$T2$	$T3$
0	1	0	1
1	1	1	1
2	1	1	1
3	1	1	0
4	1	1	0
5	0	0	1

表 2 中的矩阵也称为特征矩阵 M，该矩阵的行对应 shingle，列对应文本集合。由于特征往往非常稀疏，在处理大量文本对的相似度计算时非常不方便。

2.5. 最小哈希签名(MinHash)

通过最小哈希可以把特征矩阵转变为签名矩阵，该签名矩阵的行对应最小哈希函数，列对应文本集合。通常情况下签名矩阵所需的存储空间比特征矩阵 M 的存储空间要小很多。最小哈希签名的计算过程如算法 1 所示：

算法 1：最小哈希签名

输入：特征矩阵 M，哈希函数 $h_1,h_2,\cdots,h_n$

输出：签名矩阵 S

对于所有的每个哈希函数(i)，每一列(c)，签名矩阵中 S 的 $S(i,c)$ 表示第 i 个哈希函数及第 c 列对应的最小哈希值。

Step1 循环对特征矩阵中的每一行进行处理，计算当前行 r 的哈希值，$h_1(r),h_2(r),\cdots,h_n(r)$

Step2 对每列 c 进行如下操作

(a) 若特征矩阵中 $M(i,c)=0$，则什么都不做；

(b) 若 $M(i,c)=1$，那么对于每个 $i=1,2,\cdots,n$，将 $S(i,c)$ 设为原来 $S(i,c)$ 和 $h_i(r)$ 之中的最小值。

若 $h_1=(x+1)\bmod 7, h_2=(3x+1)\bmod 7$，则经过两个哈希函数变换之后的特征矩阵 M 见表 3，签名矩阵计算结果见表 4，最后通过签名矩阵计算得到的 Jaccard 相似度见表 5。

Table 3. Characteristic matrix M after two hash function transformations
表 3. 经过两个哈希函数变换之后的特征矩阵 M

shingles	$T1$	$T2$	$T3$	$h_1=(x+1)\bmod 7$	$h_2=(3x+3)\bmod 7$
0	1	0	1	1	3
1	1	1	1	2	6
2	1	1	1	3	2
3	1	1	0	4	5
4	1	1	0	5	1
5	0	0	1	6	4

Table 4. Signature Matrix S
表 4. 签名矩阵 S

Hash 函数	$T1$	$T2$	$T3$
h_1	1	2	1
h_2	1	1	2

Table 5. Jaccard similarity of text pairs
表 5. 文本对的 Jaccard 相似度

	$JSim(T1,T2)$	$JSim(T1,T3)$	$JSim(T2,T3)$
真实 Jaccard 相似度	4/5	3/6	2/6
通过签名估计的 Jaccard 相似度	1/2	1/2	0

2.6. LSH

若有 N 个文本，进行两两相似度将需要比较 $N^2/2$ 次。这将极大增加计算量。LSH 仅仅关注散列到同一个桶中的候选对，只检查这些候选对之间的相似度，它的基本思想就是将签名矩阵划分成 b 个行条，每个行条由 r 行组成。其中 $n = br$，n 为哈希函数的个数，即签名矩阵的行数。行条化策略能够使得相似列比不相似列更有可能成为候选对，从而大大减少文本对之间的比较次数。

2.7. 基于 LSH 技术的试题相似度检测算法

本文具体处理的过程见算法 2。

算法 2：基于 LSH 技术的试题相似度检测算法

输入：试题文本集合 $T_1,T_2,\cdots,T_N$，Jaccard 相似度阈值 T^*

输出：相似试题文本对及相应的 Jaccard 估计相似度

Step1 对所有试题文本进行中文分词，同时去除停用词，删除首尾空格

Step2 把上一步中的结果转换成 K-shingle 集，得到特征矩阵 M

Step3 随机生成 n_hashes 个哈希函数 $h_1,h_2,\cdots,h_{n\_hashes}$，根据算法 1 计算出签名矩阵 S

Step4 通过 LSH 得到所有的候选相似对，具体是将签名矩阵划分成 n_bands 个行条，每个行条由 $r=N/n\_bands$ 行组成，每个行条采用一个哈希函数对 r 个整数构成的列向量进行散列。若两向量相同，则会散列到同一个桶中，从而得到候选相似对。

Step5 计算所有候选相似对的 Jaccard 相似度，若大于相似度阈值 T^*，则两试题文本相似。

3. 实验结果及分析

本文采用的数据为《数据库课程》试题库，其中，判断题 100 道，填空题 100 道，选择题 100 道，解答题 50 道，综合题 20 道。考虑到组卷系统的大众化使用，更方便的在任何地方组卷，不借助除了 Office 之外的其它任何组件，所有功能全部在 VBA 中实现。程序采用了 Word 自带的中文分词功能，对试题内容去除停用词、删除空格。然后转换为 K-shingle 集，从而把所有的试题内容转换成了特征矩阵 M，通过最小哈希(Minhash)计算出试题签名，最后通过 LSH 快速的找出候选相似对，同时计算出相应的 Jaccard 相似度。若该相似度大于给定的阈值，则输出。同时在组卷的过程中避免了相似试题，从而由此提高组卷的质量。

在实验中，设置题 3 与题 5 相似，题 7 与题 9 相似，题 269 与题 271 相似。哈希函数个数 n_hashes 为 20 个，LSH 中的行条数 n_bands 为 4 个。运行结果如表 6 所示，其中的真实相似度是在不同 Shingle 长度下，对两 Shingle 集计算的 Jaccard 相似度。估计相似度是对两试题的最小哈希签名计算的 Jaccard 相似度。

Table 6. Results of different shingle lengths
表 6. 不同 shingle 长度的程序运行情况

Shingle-Size	题 3 与题 5		题 7 与题 9		题 269 与题 271		总耗时
	真实相似度	估计相似度	真实相似度	估计相似度	真实相似度	估计相似度	
1	0.92	0.8	0.71	0.48	0.89	0.74	30.3S
2	0.8	0.67	0.67	0.43	0.87	0.67	31.33S
3	0.67	0.48	0.6	0.48	0.84	0.6	31.33S

表 6 表明：由于试题文本一般不会太长，所以 shingle 长度不宜过大，设置为 2 比较合适。

当改变哈希函数 n_hashes 的数目时，运行情况表 7 所示。

表 7 表明：当哈希函数数目和行条数目较少时，真实度与近似相似有较大差距，且会漏检；随着哈希数目和行条数目的增多，差距越来越小。漏检率越来越低，精度越来越高。

组卷情况统计如图 1 所示。

Table 7. Shingle length is 2, the result of different hash functions and bands
表 7. Shingle 长度为 2，不同哈希函数和行条数目的运行情况

n_hashes	*n_bands*	题 3 与题 5 签名相似度	题 7 与题 9 签名相似度	题 269 与题 271 签名相似度	候选相似对检测情况	总耗时
20	4	0.67	未发现	0.67	发现 3 对(包括 2 对目标对)	31.3S
100	20	0.72	0.57	0.82	发现 5 对(包括 3 对目标对)	31.6S
200	40	0.72	0.58	0.878	发现 8 对(包括 3 对目标对)	32.8S
400	80	0.76	0.59	0.90	发现 15 对(包括 3 对目标对)	36.5S

试卷A

题型号	题型	题库名	所在位置	备注（是否雷同）
5	判断题	数据库基础	61	
5	判断题	数据库基础	75	
5	判断题	数据库基础	105	
5	判断题	数据库基础	99	
5	判断题	数据库基础	65	
1	填空题	数据库基础	29	
1	填空题	数据库基础	47	
1	填空题	数据库基础	33	
1	填空题	数据库基础	17	
1	填空题	数据库基础	9	相似试题有：7,相似度.587301587301587;
1	填空题	数据库基础	45	
1	填空题	数据库基础	3	相似试题有：5,相似度.762114537444934;
4	选择题	数据库基础	197	
4	选择题	数据库基础	135	
4	选择题	数据库基础	139	
4	选择题	数据库基础	205	相似试题有：211,相似度.590457256461233;
4	选择题	数据库基础	129	
4	选择题	数据库基础	127	
4	选择题	数据库基础	195	相似试题有：151,相似度.577909270216962;
4	选择题	数据库基础	131	
4	选择题	数据库基础	207	
4	选择题	数据库基础	201	
2	简答题	数据库基础	239	
2	简答题	数据库基础	235	
2	简答题	数据库基础	231	
2	简答题	数据库基础	237	
2	简答题	数据库基础	221	
2	简答题	数据库基础	233	
3	综合题	数据库基础	261	
3	综合题	数据库基础	267	
3	综合题	数据库基础	251	
3	综合题	数据库基础	253	相似试题有：249,相似度.822323462414579; 265,相似度.754385964912281;

试卷B

题型号	题型	题库名	所在位置	备注（是否雷同）
5	判断题	数据库基础	95	
5	判断题	数据库基础	87	
5	判断题	数据库基础	81	
5	判断题	数据库基础	111	
5	判断题	数据库基础	89	
1	填空题	数据库基础	31	
1	填空题	数据库基础	21	
1	填空题	数据库基础	13	
1	填空题	数据库基础	7	相似试题有：9,相似度.587301587301587;
1	填空题	数据库基础	11	
1	填空题	数据库基础	25	
1	填空题	数据库基础	15	
4	选择题	数据库基础	155	
4	选择题	数据库基础	121	
4	选择题	数据库基础	147	
4	选择题	数据库基础	153	
4	选择题	数据库基础	123	
4	选择题	数据库基础	137	
4	选择题	数据库基础	115	
4	选择题	数据库基础	217	
4	选择题	数据库基础	133	
4	选择题	数据库基础	141	
2	简答题	数据库基础	227	
2	简答题	数据库基础	225	
2	简答题	数据库基础	219	
2	简答题	数据库基础	229	
2	简答题	数据库基础	223	
2	简答题	数据库基础	239	与A卷相同
3	综合题	数据库基础	249	相似试题有：253,相似度.822323462414579;
3	综合题	数据库基础	245	
3	综合题	数据库基础	265	相似试题有：253,相似度.754385964912281;
3	综合题	数据库基础	269	相似试题有：271,相似度.913875598086124;

Figure 1. The result of generating test paper
图 1. 组卷情况

4. 结语

本文提出的基于 LSH 技术的试题相似度检测方法，实验结果表明，通过该方法检测能有效减少相似试题的重复率，并极大地减少相似对的比较次数，降低时间复杂度。

基金项目

楚雄师范学院校级教改项目，项目编号：1807。

参考文献

[1] Muskan, K.M. (2017) Identifying Influential Segments from Word Co-Occurrence Networks Using AHP. *Cognitive Systems Research*, S138904171630198X.

[2] Pawar, A. and Mago, V. (2018) Calculating the Similarity between Words and Sentences Using a Lexical Database and Corpus Statistics.

[3] Abujar, S., Hasan, M. and Hossain, S.A. (2019) Sentence Similarity Estimation for Text Summarization Using Deep Learning. In: Kulkarni, A., Satapathy, S., Kang, T. and Kashan, A., Eds., *Proceedings of the 2nd International Conference on Data Engineering and Communication Technology*, Advances in Intelligent Systems and Computing, Vol. 828, Springer, Singapore. https://doi.org/10.1007/978-981-13-1610-4_16

[4] Chen, Q., Hu, Q.M., Huang, X.J. and He, L. (2018) CAN: Enhancing Sentence Similarity Modeling with Collaborative and Adversarial Network. 815-824. https://doi.org/10.1145/3209978.3210019

[5] Chi, Z. and Zhang, B. (2018) A Sentence Similarity Estimation Method Based on Improved Siamese Network. *Journal*

of Intelligent Learning Systems and Applications, **10**, 121-134. https://doi.org/10.4236/jilsa.2018.104008

[6] Yao, H., Liu, H. and Zhang, P. (2018) A Novel Sentence Similarity Model with Word Embedding Based on Convolutional Neural Network. *Concurrency and Computation*: *Practice and Experience*, **30**, e4415. https://doi.org/10.1002/cpe.4415

[7] Quan, Z., Wang, Z., Le, Y., Yao, B., Li, K. and Yin, J. (2019) An Efficient Framework for Sentence Similarity Modeling. *IEEE*/*ACM Transactions on Audio*, *Speech*, *and Language Processing*, **27**, 853-865. https://doi.org/10.1109/TASLP.2019.2899494

[8] Le, Y.Q., Wang, Z.-J., Quan, Z., He, J.W. and Yao, B. (2018) ACV-Tree: A New Method for Sentence Similarity Modeling. *IJCAI*, 4137-4143.

[9] 梁圣. 基于 RNN 的试题相似度计算模型研究与实现[J]. 数码设计, 2018, 7(1): 21-23.

[10] 田星, 郑瑾, 张祖平. 基于词向量的 Jaccard 相似度算法[J]. 计算机科学, 2018, 45(7): 192-195.

[11] Chen, Q., Hu, Q., Huang, J.X. and He, L. (2018) CA-RNN: Using Context-Aligned Recurrent Neural Networks for Modeling Sentence Similarity. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, New Orleans, LA, 2-7 February 2018.

[12] Leskovec, J., Rajaraman, A. and Ullman, J.D. (2015) Mining of Massive Datasets. 2nd Edition. Posts & Telecom Press, Beijing, 56-70. https://doi.org/10.1017/CBO9781139924801

[13] Manaa, M.E. and Abdulameer, G. (2018) Web Documents Similarity Using K-Shingle Tokens and MinHash Technique. *Journal of Engineering and Applied Sciences*, **13**, 1499-1505.