



## 多重检验加权融合的短文本相似度计算方法

石彩霞, 李书琴, 刘 斌

(西北农林科技大学 信息工程学院, 陕西 杨凌 712100)

**摘 要:** 传统相似度计算方法仅考虑文本结构特征或语义信息, 从而导致准确率较低。结合短文本特征稀疏的特性, 提出一种多重检验加权融合的短文本相似度计算方法 MCWFS。使用基于改进编辑距离、考虑词频、基于 Word2vec 与 LSTM 的 3 种方法分别计算相似度, 对满足多重检验标准的文本进行加权因子线性融合, 以避免因一种相似度值过大或过小导致加权相似度值异常的问题。在此基础上, 通过加权融合计算短文本相似度, 使得计算结果更加准确合理。实验结果表明, 相比层层检验和无检验融合方法, MCWFS 方法的平均准确率分别提高 16.01% 和 7.39%, 且其 F1 值可达 70.21%。

**关键词:** 短文本相似度; 多重检验加权融合; 编辑距离; 语义信息; 词频

开放科学(资源服务)标志码(OSID):



**中文引用格式:** 石彩霞, 李书琴, 刘斌. 多重检验加权融合的短文本相似度计算方法[J]. 计算机工程, 2021, 47(2): 95-102.

**英文引用格式:** SHI Caixia, LI Shuqin, LIU Bin. Method for calculating short text similarity using multi-check weighted fusion[J]. Computer Engineering, 2021, 47(2): 95-102.

## Method for Calculating Short Text Similarity Using Multi-Check Weighted Fusion

SHI Caixia, LI Shuqin, LIU Bin

(College of Information Engineering, Northwest A&F University, Yangling, Shaanxi 712100, China)

**[Abstract]** Most of the existing similarity calculation methods consider only the text structure features or the semantic information, and thus reduce the accuracy. To address the problem, this paper proposes a method, MCWFS, for calculating short text similarity using multi-check weighted fusion to deal with the sparse features of short texts. The method calculates the similarity by using three methods: similarity calculation based on improved edit distance, semantic similarity calculation considering word frequency, and similarity calculation based on the Word2vec and LSTM. Then weighted linear fusion is performed for texts that satisfy multi-check standards to avoid abnormal weighted similarity value caused by a single too large or too small similarity value. On this basis, weighted fusion is used to calculate the short text similarity to make the result more accurate and reasonable. Experimental results show that compared with the layer-by-layer check and non-check fusion methods, the proposed MCWFS method improves the average accuracy by 16.01% and 7.39% respectively, and its F1 value reaches 70.21%.

**[Key words]** short text similarity; multi-check weighted fusion; edit distance; semantic information; word frequency

**DOI:** 10.19678/j.issn.1000-3428.0056847

### 0 概述

文本相似度计算是文本处理领域的关键性技术, 广泛应用于自然语言处理(Natural Language Processing, NLP)的信息检索、文本分类、自动问答和文本重复性检测等多种任务中<sup>[1]</sup>。近年来, 文本相似度计算

在电子商务、新闻推送、推荐系统等热门领域也受到极大关注<sup>[2]</sup>。

目前, 文本相似度计算方法主要分为三类, 一是基于字符串的计算方法, 如通过统计文本共有字词数量计算相似度的 N-gram<sup>[3]</sup>和 Jaccard<sup>[4]</sup>算法, 二是基于语料库的计算方法, 如忽略词序、句法结构等关

**基金项目:** 中国博士后科学基金(2017M613216); 陕西省自然科学基金(2017JM6059); 陕西省重点研发计划(2019ZDLNY07); 陕西省博士后基金(2016BSHEDZZ121)。

**作者简介:** 石彩霞(1993—), 女, 硕士研究生, 主研方向为智能信息系统; 李书琴(通信作者), 教授、博士生导师; 刘 斌, 副教授、博士。

**收稿日期:** 2019-12-10 **修回日期:** 2020-02-05 **E-mail:** stone\_xx@163.com

键性要素利用词向量<sup>[5]</sup>基于词袋模型计算相似度的 VSM<sup>[6]</sup>和 LSA<sup>[7]</sup>等,三是基于深度学习的计算方法,如基于深度学习语义匹配模型的 DSSM<sup>[8]</sup>、通过神经网络生成词向量以计算相似度的 Word2vec<sup>[9]</sup>和 Glove<sup>[10]</sup>等。文献[11]基于 CNN 并引入多注意力机制,通过关注词汇间和句子间的语义信息来计算句子相似度。文献[12]提出一种 Siamese LSTM 方法,其利用记忆单元使 LSTM 能够存储长序列信息,从而解决 RNN 长期依赖的问题。

现有多数文本相似度计算相关研究仅考虑单一文本特征而进行相似度计算。马慧芳等人<sup>[13]</sup>融合词项共现距离相关度和类别特征来计算短文本相似度。邓涵等人<sup>[14]</sup>从句子的结构角度对句法和依存关系进行分析计算。YANG 等人<sup>[15]</sup>将浅层句法结构化特征用依赖树表示并计算相似度,但该方法不能分析句子的深层语义信息。张小川等人<sup>[16]</sup>融合主题相似度因子和词语共现度因子提出基于 LDA 的短文本相似度计算算法,但其未考虑文本的语义、词序和主题关联性等特征。

短文本具有句子简短、词语较少、语义丰富和特征稀疏的特点,本文综合考虑短文本的多种特征因素,如共现词、词频及语义信息等对文本相似度的影响,在分析传统文本相似度计算方法的基础上,利用基于深度学习的方法计算相似度,通过阈值对相似度值进行检验筛选,并将改进的 Damerau-Levenshtein 距离算法、考虑词频的语义相似度计算算法、基于 Word2vec 与 LSTM 的相似度计算算法 3 种考虑单因素的计算方法进行加权融合,应用于短文本相似度计算,从而使得计算结果更加准确合理。

## 1 相关工作

### 1.1 改进的 Damerau-Levenshtein 距离算法

Levenshtein 距离<sup>[17]</sup>于 1965 年由苏联数学家 Vladimir Levenshtein 提出,其又被称为编辑距离(Edit Distance),主要用于比较 2 个字符串的相似度。Levenshtein 距离是指将一个字符串序列通过插入、删除和替换等单字符操作转变为另一个字符串所需的最小操作数量。

短文本具有句子简短的特点,相似文本之间会有较多的共现词,适合通过编辑距离计算相似度。Frederickj Damau 提出了改进 Levenshtein 距离的 Damerau-Levenshtein 距离<sup>[18]</sup>,其考虑置换操作对编辑距离的影响,但本质依然是编辑距离。Damerau-Levenshtein 距离的计算方式如下:

设有 2 个字符串  $S$  和  $T$ ,其中, $S$  为长度为  $m$  的源字符串, $T$  为长度为  $n$  的目标字符串,用  $dl_{s,t}$  表示  $S$  和  $T$  之间的 Damerau-Levenshtein 距离,则可以构造  $(m+1) \times (n+1)$  阶矩阵  $D_{i,j} = D(s_1, s_2, \dots, s_i, t_1, t_2, \dots, t_j)$ ,  $0 \leq i \leq m$ ,

$0 \leq j \leq n$ ,通过式(1)来计算 2 个字符串之间的 Damerau-Levenshtein 距离:

$$dl_{s,t}(i,j) = \begin{cases} \max(i,j), \min(i,j)=0 \\ \min \left( dl_{s,t}(i-1,j)+1, dl_{s,t}(i,j-1)+1, \right. \\ \quad \left. dl_{s,t}(i-1,j-1)+1(s_i \neq t_j), dl_{s,t}(i-2,j-2)+1 \right), \\ \quad i,j>1, s_i=t_{j-1} \text{ 且 } s_{i-1}=t_j \\ \min \left( dl_{s,t}(i-1,j)+1, dl_{s,t}(i,j-1)+1, \right. \\ \quad \left. dl_{s,t}(i-1,j-1)+1(s_i \neq t_j) \right), \text{ 其他} \end{cases} \quad (1)$$

由式(1)计算得到的 Damerau-Levenshtein 是一个正整数,如果用其衡量相似度,将缺少一个限定值作为界定是否相似的标准,因此,本文提出 DLR (Damerau-Levenshtein-Ratio),其将 2 个文本的编辑距离转化为比值形式,通过式(2)计算 DLR 以表示 2 个文本之间的相似度:

$$DRL_{s,t} = 1 - \frac{dl_{s,t}}{L_{\max}} \quad (2)$$

其中, $L_{\max}$  表示字符串  $S$  和  $T$  长度的最大值。DLR 的计算过程伪代码如算法 1 所示。

#### 算法 1 Damerau-Levenshtein-Ratio 计算算法

输入 短文本  $S = (s_1, s_2, \dots, s_i, \dots, s_m)$ ,  $0 \leq i \leq m$ , 短文本  $T = (t_1, t_2, \dots, t_j, \dots, t_n)$ ,  $0 \leq j \leq n$

输出 相似度值 DLR

1. 初始化矩阵第 1 列,  $D[i][0]=i$ ;
2. 初始化矩阵第 1 行,  $D[0][j]=j$ ;
3. If  $s[i]==t[j]$ , cost = 0;
4. Else cost = 1;
5. End;
6.  $D[i][j] = \min(D[i-1][j]+1(\text{删除}), D[i][j-1]+1(\text{插入}), D[i-1][j-1]+cost(\text{替换}))$ ;
7. If  $i>1$  and  $j>1$  and  $s[i]==t[j-1]$  and  $s[j-1]==t[i]$ ;
8.  $D[i][j] = \min(D[i][j], D[i-2][j-2]+cost(\text{置换}))$ ;
9. End;
10.  $DL = D[m][n]$ ;
11.  $L_{\max} = \max(m, n)$ ;
12. Return  $DLR = 1 - \frac{dl_{s,t}}{L_{\max}}$

### 1.2 考虑词频的语义相似度计算算法

传统基于句子形态的文本相似度计算算法<sup>[19]</sup>利用词形、句长和词序特征计算相似度,但它们忽略了文本的语义信息,导致效果不佳且对短文本不适用。短文本会因词语数量较少而引起语义稀疏性,因此,本文在计算语义信息的基础上,通过词频对词语赋予权重,并且考虑未收录词和文本语义不全对相似度的影响,分别计算只有关键词和含有非关键词 2 种情况下的文本相似度,取两者较大值作为文本的相似度值。

定义  $S_1, S_2$  为 2 个待计算相似度的句子,令  $KS_1 = \{w_1, w_2, \dots, w_m\}$  为句子  $S_1$  的关键词集合,其中,  $w_k$  表示从  $S_1$  中提取到的关键词,同样地,  $KS_2 = \{w_1, w_2, \dots, w_n\}$  为句子  $S_2$  的关键词集合,则利用传统基于知网知识库的方式计算句子  $S_1$  和  $S_2$  关键词的语义相似度如下:

$$KSim(S_1, S_2) = \frac{m \cdot WSim(KS_1, KS_2) + n \cdot WSim(KS_2, KS_1)}{m + n} \quad (3)$$

短文本具有句短词少的特点,使得词频大的词语对句子相似度计算有较大影响,因此,本文根据词频对词语赋予权重,句子  $S_1$  中的词语  $w_1$  对句子  $S_2$  的相似度为  $w_1$  对  $S_2$  中全部词语相似度的最大值。改进后  $S_1$  对  $S_2$  的关键词集合的相似度计算如式(4)、式(5)所示:

$$WSim(S_1, S_2, tf) = \frac{\sum_{i=1}^m tf_{i1} \cdot \text{word\_sim}_{w_i \in S_1}(w_i, S_2)}{\sum_{i=1}^m tf_{i1}} \quad (4)$$

$$\text{word\_sim}_{w_i \in S_1}(w_i, S_2) = \max(\text{sim}(w_i, w_j), j=1, 2, \dots, n) \quad (5)$$

其中,  $tf_i$  表示词语  $i$  在短文本中的词频。

在知网知识库中,一个词语具有多个义项,而每个义项又由多个义原组成。知网中 2 个词语的义项相似度是通过第一基本义原、其他义原、关系义原和关系符号义原组合后加权表示的<sup>[20]</sup>,义项相似度计算如式(6)所示。假设词语  $w_1$  和词语  $w_2$  的义项分别表示为  $w_1 = \{s_{11}, s_{12}, \dots, s_{1m}\}$ 、 $w_2 = \{s_{21}, s_{22}, \dots, s_{2n}\}$ ,则词语  $w_1$  和词语  $w_2$  的相似度为词语  $w_1$  和词语  $w_2$  义原相似度的最大值,如式(7)所示。

$$\text{sim}(s_1, s_2) = \sum_{i=1}^4 \beta_i \prod_{j=1}^i \text{sim}_j(s_1, s_2) \quad (6)$$

$$\text{sim}(w_1, w_2) = \max_{\substack{i=1, 2, \dots, m, \\ j=1, 2, \dots, n}} \text{sim}(s_{i1}, s_{j2}) \quad (7)$$

在式(6)中,  $\beta_1 \geq \beta_2 \geq \beta_3 \geq \beta_4$ ,  $\text{sim}(s_1, s_2)$  表示义项  $s_1$  和义项  $s_2$  的第一义原相似度,以此类推。4 种因子按照文献[21]取值,  $\beta_1 = 0.4, \beta_2 = 0.3, \beta_3 = 0.2, \beta_4 = 0.1$ 。吴健等人<sup>[22]</sup>认为节点深度对义原相似度有一定影响,通常通过 2 个义原之间的路径距离来计算义原相似度,如下:

$$\text{sim}(p_1, p_2) = \frac{\alpha}{d + \alpha} \quad (8)$$

其中,  $d$  表示 2 个义原在层次结构中的路径距离,  $\alpha$  是调节参数,取值为 1.6。

令句子  $S_1$  和句子  $S_2$  的相似度  $KSim$  为  $KSim$  和  $TSim$  的最大值,如式(9)所示。 $KSim$  表示仅有关键词时句子的相似度,计算公式如式(10)所示,  $TSim$  表示含有非关键词时句子的相似度,计算公式如

式(11)所示。 $TSim$  的计算方式与  $KSim$  相同,只是两者参与计算的词语数量不同。

$$KSim(S_1, S_2, w_1, w_2) = \max(KSim(S_1, S_2), TSim(S_1, S_2)) \quad (9)$$

$$KSim(S_1, S_2) = \left( \frac{WSim(KS_1, KS_2, w_1)}{m} + \frac{WSim(KS_2, KS_1, w_2)}{n} \right) / 2 \quad (10)$$

$$TSim(S_1, S_2) = \left( \frac{WSim(TS_1, TS_2, w_1)}{m} + \frac{WSim(TS_2, TS_1, w_2)}{n} \right) / 2 \quad (11)$$

### 1.3 基于 Word2vec 与 LSTM 的相似度计算算法

Word2vec 即为词向量,是由 Google 公司开源的一款深度学习模型<sup>[9]</sup>,作用是将自然语言中的词语或者文字转化为计算机可以理解和处理的向量形式。Word2vec 可分为 CBOW (Continue Bag-of-Word) 和 Skip-gram 2 种模型。CBOW 输入某个特征词的上下文关联词的词向量,输出该特征词的词向量,而 Skip-gram 相反,其输入特征词对应的词向量,输出特征词的上下文对应词的词向量。

HOCHREITER 等人<sup>[23]</sup>提出长短期记忆(Long Short-Term Memory, LSTM)网络,其为一种特殊的 RNN,使用记忆单元替换原 RNN 中的隐层节点,以达到学习长期依赖信息的目的。LSTM 的计算过程如式(12)~式(17)所示,其中,式(12)和式(13)表示更新记忆细胞的状态,式(14)~式(16)分别为增加历史信息的输入门、遗忘历史信息的遗忘门和输出门,式(17)表示利用  $\tanh$  作用于当前记忆细胞状态,再由输出门输出最后信息。

$$\tilde{c}_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (12)$$

$$c_t = f_t c_{t-1} + i_t \tilde{c}_t \quad (13)$$

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (14)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (15)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o) \quad (16)$$

$$h_t = o_t \tanh(c_t) \quad (17)$$

其中,  $W_{xy}$  为权重矩阵,表示从神经元  $x$  到  $y$  的权重,  $c_t$  表示当前细胞状态,  $\tilde{c}$  表示候选值,  $x$  表示记忆单元的输入,  $h$  表示输出,  $b$  表示偏置权重。

Skip-gram 相较于 CBOW 模型有更高的语义准确率,其可以通过跳跃词汇来构造词组,避免因窗口大小限制而丢失文本的语义信息。因此,本文采用 Skip-gram 模型作为训练框架,训练窗口设置为 5,词向量维度设置为 100。以 LSTM 为基础,由输入层将输入的句子按照单个词语在词典中的特定位置关系得到编号序列,嵌入层利用 Word2vec 模型将由输入层得到的词语编号序列映射为词向量,再将得到的词向量作为 LSTM 层的输入,将 LSTM 层输出的结



果进行拼接并作为全连接层的输入,利用 Dropout 防止过拟合,最终由输出层输出 2 个文本的相似度值。基于 Word2vec 与 LSTM 的相似度计算模型结构如图 1 所示。

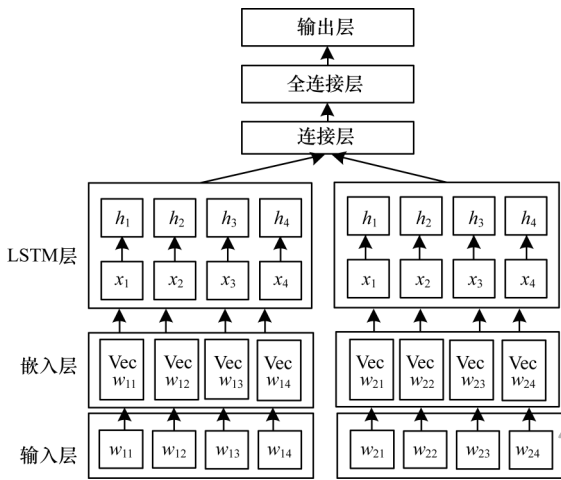


图 1 基于 Word2vec 与 LSTM 的相似度计算模型结构  
Fig.1 Structure of similarity calculation model based on Word2vec and LSTM

2 多重检验加权融合的相似度计算

基于改进 Damerau-Levenshtein 距离的相似度计算(DLRSim)算法考虑 2 个短文本之间的词序和共有词,从短文本句短词少的结构特征角度计算相似度;基于语义的相似度计算(KTSim)算法为解决短文本的特征稀疏性问题,加入词频并考虑关键词和非关键词对短文本相似度的影响,结合知网义原信息从词语的义原层面计算句子的相似度;基于 Word2vec 与 LSTM 的相似度计算(WLSim)算法从深度学习的角度构建模型,将文本转化为词向量然后对语义等特征进行学习从而计算相似度。

为将上述 3 种相似度计算算法进行有效融合并应用于短文本的相似度计算,本文提出多重检验加权融合的相似度计算(MCWFS)方法,通过实验对以上 3 种相似度计算方法分别确定一个相似度阈值,当 2 个文本间的 3 个相似度值中至少有 2 个大于对应阈值时,认为两者可能相似,则进行加权融合以及相似度计算,否则认为两者不相似。上述操作可以避免文献[24]中因一种相似度小于阈值而被认为不相似从而无法参与下一阶段运算的情况,最终使得相似度计算结果更加准确。本文通过线性加权融合方法计算相似度值的公式如下:

$$FSsim = \alpha \cdot DLRSim + \beta \cdot KTSim + \gamma \cdot WLSim \quad (18)$$

其中,权重因子满足  $\alpha + \beta + \gamma = 1$ ,通过实验调节权重因子从而确定它们的最佳取值组合。

相似度计算过程伪代码如算法 2 所示,MCWFS 方法的计算流程如图 2 所示。

算法 2 相似度计算算法

输入 短文本  $S = (s_1, s_2, \dots, s_i, \dots, s_m), 0 \leq i \leq m$ , 短文本  $T = (t_1, t_2, \dots, t_j, \dots, t_n), 0 \leq j \leq n$   
输出 2 个文本的相似度值 FSim  
1. 文本数据预处理;  
2. For n = 1 to 3;  
3. 计算 S 和 T 的 DLRSim;  
4. 计算仅含关键词的 KSim;  
5. 计算含非关键词的 TSim;  
6. 令 KTSim = max(KSim, TSim);  
7. 计算 WLSim;  
8. 按照相似度阈值  $t_1, t_2, t_3$  进行多重检验;  
9. If 至少有 2 种相似度值大于等于对应阈值;  
10. 加权计算 FSim;  
11. Else FSim=0;  
12. End;  
13. End;  
14. Return FSim

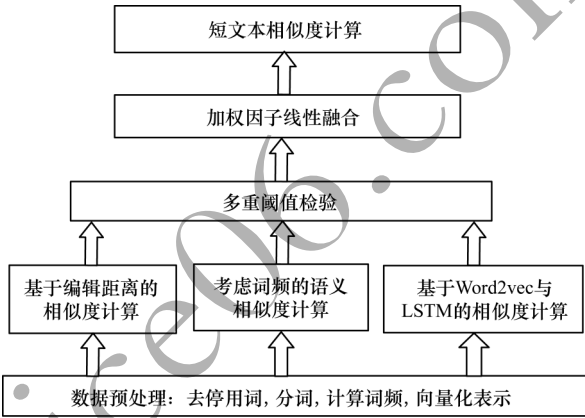


图 2 MCWFS 方法计算流程  
Fig.2 Calculation flow of MCWFS method

3 实验结果与分析

3.1 实验数据

为验证本文方法的有效性,采用蚂蚁金融 NLP 挑战数据集([https://pan.baidu.com/s/1yEeThJi\\_HHxwQjbrG3OArQ](https://pan.baidu.com/s/1yEeThJi_HHxwQjbrG3OArQ)),该数据集信息如表 1 所示,每行由序号、2 句短文本和 1 个相似性标志位组成,共包含 102 477 组句子对。首先对数据集进行预处理,去除格式不正确的句子对,采用百度停用词表对文本去除停用词,并使用 HanLP 工具包进行分词并统计词频。在预处理后剩余的 102 373 组句子对中,正样本为 18 668 组,负样本为 83 705 组。本文选取 80% 的数据作为训练集,20% 的数据作为测试集,以进行模型训练。

表 1 实验数据集信息

Table 1 Experimental dataset information			
序号	文本 1	文本 2	相似性标志
1	花呗借款要不要利息	用花呗付款要利息吗	1
2	花呗最低额度是多少	我的花呗可用额度是多少	0

3.2 实验环境与度量标准

本文实验环境为 Windows7 操作系统,使用 MyEclipse 作为开发工具,数据库采用 MySQL5.6 版本,使用 Java 和 Python2.7 开发语言实现本文相似度计算方法,开发环境采用 JDK1.8。

在结果分析中,本文主要采用 F1 值作为 3 种检验阈值的选择标准,同时采用文本信息处理领域常用的召回率 (Recall) 和准确率 (Accuracy) 作为相似度计算质量评价指标,将本文方法分别与传统方法、无检验的相似度计算方法、未融合的相似度计算方法进行比较,最后通过 Python 将实验对比结果进行可视化展示。在评价指标中,准确率是指所有被正确预测为正和负的样本数占总样本数的概率,精确率 (Precision) 是指正确预测为正的样本占全部预测为正的样本的比例,召回率是指在实际为正的样本中被预测为正样本的概率,F1 值是精确率和召回率的加权调和平均。4 种指标的计算公式分别如式 (19)~式 (22) 所示:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \times 100\% \tag{19}$$

$$Precision = \frac{TP}{TP + FP} \times 100\% \tag{20}$$

$$Recall = \frac{TP}{TP + FN} \times 100\% \tag{21}$$

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \times 100\% \tag{22}$$

其中,TP 表示实例是正类实际也被预测为正类的样本数量,TN 表示实例是负类实际也被预测为负类的样本数量,FN 表示实例是正类实际被预测为负类的样本数量,FP 表示实例是负类实际被预测为正类的样本数量。

3.3 结果分析

3.3.1 检验阈值选择

实验通过调节相似度阈值,对比 DLRSim 算法、KTSim 算法和 WLSim 算法的 F1 值,从而确定 3 种相似度的检验阈值标准,结果如图 3 所示。从图 3 可以看出,在 F1 值取值最大时,对于 DLRSim 算法、

KTSim 算法和 WLSim 算法,相似度阈值分别取 0.40、0.42 和 0.47。因此,本文将 3 种检验阈值分别设置为  $t_1=0.40$ 、 $t_2=0.42$ 、 $t_3=0.47$ 。

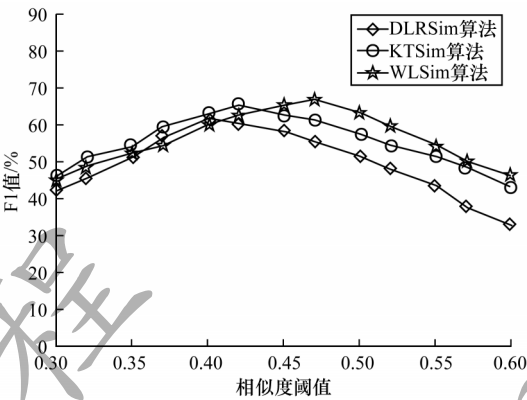


图 3 不同算法的 F1 值对比结果

Fig.3 Comparison results of F1 values of different algorithms

3.3.2 加权因子调节

为对满足多重检验标准的文本进行加权融合相似度计算,本文运用控制变量法对加权融合的加权因子进行调节,选择相似度阈值取不同值时的召回率作为评价指标,调整  $\alpha$ 、 $\beta$  和  $\gamma$  的取值组合,通过观察召回率来确定参数的最佳取值组合。由于实验数据较多,表 2 选取相似度阈值为 0.42、0.44、0.46、0.48 和 0.50 时的数据进行展示。从表 2 可以看出:减小 DLRSim 算法所占权重,召回率会明显增大,这是因为 DLRSim 算法主要从文本的编辑距离角度计算相似度,影响因素主要为共有词,因此其占比相对较小;增大 KTSim 算法的加权因子,召回率有所提升,因此,基于语义考虑词语义项的 KTSim 算法对召回率影响较大;基于深度学习模型的 WLSim 算法具有学习能力,可有效保持对历史信息的较长记忆,获取整个文本的语义特征信息,在序列化数据处理时有一定优势,因此,其影响力更大一些。通过对比最终确定加权因子取值为: $\alpha=0.21$ 、 $\beta=0.36$ 、 $\gamma=0.43$ 。

表 2 不同加权因子取值组合下的召回率对比结果

Table 2 Comparison results of recall under different combination of weighted factors

$\alpha$	$\beta$	$\gamma$	Recall@0.42 /%	Recall@0.44 /%	Recall@0.46 /%	Recall@0.48 /%	Recall@0.50 /%
0.50	0.20	0.30	38.02	35.54	28.33	26.21	31.58
0.30	0.30	0.40	67.12	64.84	63.56	58.25	55.21
0.25	0.50	0.25	74.26	73.44	69.16	65.53	63.35
0.23	0.45	0.32	76.68	74.25	71.21	67.55	65.64
0.21	0.36	0.43	83.84	78.24	75.32	73.26	70.56
0.15	0.40	0.45	79.19	75.16	74.04	71.22	68.65
0.15	0.25	0.60	56.24	57.29	54.21	50.28	44.34

### 3.3.3 不同方法的实验结果对比

本节将从以下4个方面对不同方法的实验结果进行对比:

#### 1) 词频对短文本相似度的影响

为验证词频对短文本相似度的影响,将KTSim算法与文献[21]中的HowNet算法进行实验对比,选取词频不同的2组短文本,分别用2种算法计算相似度,数据结果如表3所示。从表3可以看出,对于词频不为1的文本,2种算法相似度计算结果差异较大,而对于词频都为1的文本,2种算法计算结果的差异较小,但KTSim算法的计算结果优于HowNet算法。

表3 词频对相似度计算结果的影响

Table 3 Influence of word frequency on similarity calculation results

文本1	文本2	KTSim算法	HowNet算法
我是商家,怎么开通 商家收款码	如何开通商家 收款码	0.756	0.678
关闭花呗支付功能	如何关闭花呗	0.532	0.516

#### 2) 检验方法对比

为验证本文多重检验方法的准确性,以不同阈值下的准确率作为评价标准,将本文多重检验融合(MCWFS)方法与无检验融合(NCWFS)方法和层层检验融合(ECWFS)方法进行实验对比。NCWFS即无阈值检验而直接加权计算,ECWFS中只要有一种相似度值不满足阈值则认为不相似,不进行加权运算。从图4可以看出,MCWFS方法的效果优于NCWFS方法,这是因为无检验融合方法无法对只有一种相似度值过大的情况进行筛选,从而将异常值引入运算,将实际不相似的文本判定为相似,降低了准确率。ECWFS方法的效果相对较差,原因是层层检验会因为一种相似度值不满足阈值而无法参与加权运算,使得计算中将较多相似文本判定为不相似,降低了准确率。随着阈值的增大,方法的准确率增势趋于平缓后开始下降,这是因为不相似的短文本数逐渐趋于实际值,然后又大于实际值,而判断正确的相似文本数在减少。相比ECWFS方法和NCWFS方法,MCWFS方法在准确率上平均分别提高16.01%和7.39%,即多重检验融合方法具有明显优势。

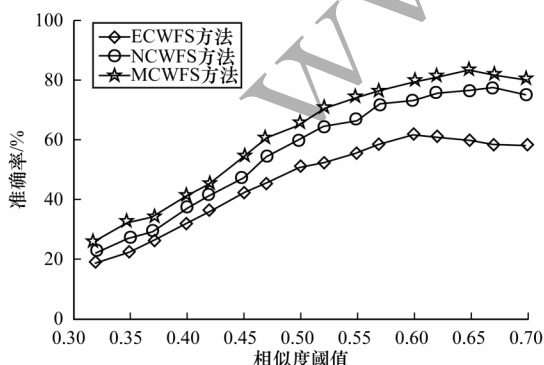


图4 3种检验方法的准确率对比结果

Fig.4 Accuracy comparison results of three test methods

#### 3) 融合方法对比

为验证加权融合方法的有效性,将其与未融合的DLRSim、KTSim和WLSim算法进行对比,以不用阈值时的召回率作为衡量标准,结果如图5所示。从图5可以看出,在相同阈值情况下,MCWFS方法的召回率最大,其具有明显优势,原因是该方法对3种对比算法所考虑的不同角度的文本特征进行了加权计算并筛选异常值,从而提高了召回率。

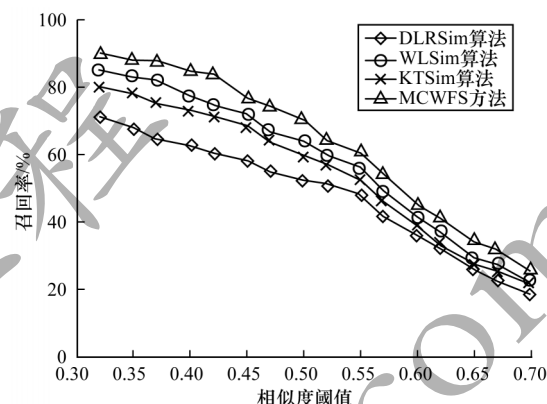


图5 融合和未融合方法的召回率对比结果

Fig.5 Comparison results of recall between fusion method and non-fusion methods

#### 4) 与传统相似度计算方法的对比

将本文多重检验加权融合计算方法与传统的Jaccard相似度计算方法、基于Word2vec的余弦相似度计算方法<sup>[25]</sup>和利用WordNet<sup>[26]</sup>计算词语和句子相似度的方法进行实验对比,以不同阈值下的F1值作为评价标准,结果如图6所示。

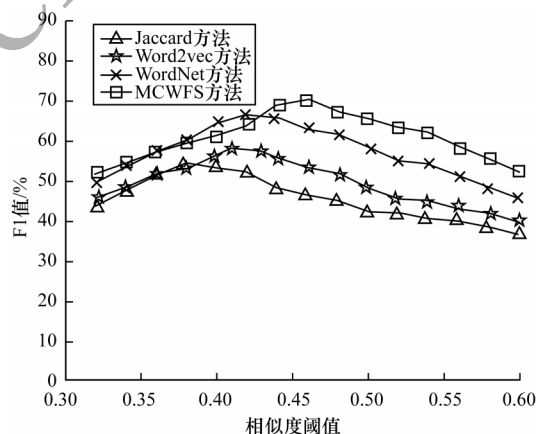


图6 不同方法的F1值对比结果

Fig.6 Comparison results of F1 values of different methods

从图6可以看出,4种方法的F1值变化趋势大致相同,MCWFS方法在相似度阈值为0.46时F1取得最大值70.21%,而Word2vec方法和WordNet方法分别在阈值为0.41和0.42时F1取得最大值,分别为58.26%和66.25%,Jaccard方法在阈值为0.38时F1取得最大值53.26%。Jaccard方法的主要影响因素为2个文本之间



的共有词,其无法利用更丰富的信息进行计算,因此,F1值最小。WordNet方法需要大规模的语料库,无法计算词库未收录的词语的相似度值,Word2vec方法虽然解决了文本的数据稀疏问题,但是其将词向量进行余弦运算,并不能代表语义关系,容易出现将相似词语较多但语义相悖的文本计算为相似文本的问题。而本

文MCWFS方法具有明显优势,因为其既考虑了文本的表型特征,也考虑了语义等信息,同时对异常值进行筛选,使相似度计算更加准确。

利用4种方法计算短文本相似度的数据结果如表4所示,从表4可以看出,本文方法计算性能最优。

表4 不同方法的相似度计算结果对比

Table 4 Comparison of similarity calculation results of different methods

序号	文本1	文本2	Jaccard方法	Word2vec方法	WordNet方法	MCWFS方法
1	花呗借款要不要利息	用花呗付款要利息吗	0.812 4	0.538 4	0.564 5	0.833 2
2	花呗最低额度是多少	我的花呗可用额度是多少	0.746 4	0.440 2	0.482 2	0.431 5

4 结束语

针对传统文本相似度计算方法准确率较低的问题,本文结合短文本句子简短、特征稀疏和语义丰富等特点,提出一种多重检验加权融合的短文本相似度计算方法。该方法通过改进编辑距离从句子表型特征计算短文本相似度,考虑文本简短对相似度的影响而加入词频作为词语权重,从语义角度计算相似度,同时利用深度学习模型进行计算,解决语义相似度对大规模语料库的依赖问题。在此基础上,将满足多重检验标准的3种相似度值进行加权融合并用于短文本相似度计算,以降低异常值对相似度值的影响并提高计算结果的准确率。实验结果表明,该方法的计算性能优于WordNet、Word2vec等方法。

受中文短文本数据集较少的影响,本文的阈值和加权因子的取值选择对数据具有依赖性和针对性,下一步将利用不同领域的数据集,通过机器学习的方式对参数进行选取。此外,基于深度学习的方式对参数进行选取。此外,基于深度学习的LSTM模型在处理短文本时具有一定优势,但对于较长文本而言其存在局限性,因此,后续考虑加入Attention机制使该模型更加灵活高效。

参考文献

[ 1 ] PANG Liang, LAN Yanyan, XU Jun, et al. A survey on deep text matching[J]. Chinese Journal of Computers, 2017, 40(4): 985-1003. (in Chinese)  
庞亮, 兰艳艳, 徐君, 等. 深度文本匹配综述[J]. 计算机学报, 2017, 40(4): 985-1003.

[ 2 ] GOMAA W H, FAHMY A A. A survey of text similarity approaches[J]. International Journal of Computer Applications, 2013, 68(13): 13-18.

[ 3 ] MAIPRADIT R, HATA H, MATSUMOTO K. Sentiment classification using N-gram inverse document frequency and automated machine learning[J]. IEEE Software, 2019, 36(5): 65-70.

[ 4 ] AYUB M, GHAZANFAR M A, MAQSOOD M, et al. A Jaccard base similarity measure to improve performance of CF based recommender systems[C]//Proceedings of 2018

International Conference on Information Networking. Washington D. C., USA: IEEE Press, 2018: 16-22.

[ 5 ] LI Xiaotao, YOU Shujuan, CHEN Wei. An algorithm of semantic similarity between words based on word single-meaning embedding model[J]. Acta Automatica Sinica, 2020, 46(8): 1654-1669. (in Chinese)  
李小涛, 游树娟, 陈维. 一种基于词义向量模型的词语语义相似度算法[J]. 自动化学报, 2020, 46(8): 1654-1669.

[ 6 ] ZHANG Peiyun, CHEN Enhong, XIE Rongjian, et al. Computation of document similarity based on metadata and domain concept tree[J]. Systems Engineering and Electronics, 2014, 36(3): 189-195. (in Chinese)  
张佩云, 陈恩红, 谢荣见, 等. 基于元数据与领域概念树的文本相似度计算[J]. 系统工程与电子技术, 2014, 36(3): 189-195.

[ 7 ] KAUR H, MAINI R. Granularity-based assessment of similarity between short text strings[C]//Proceedings of the 3rd International Conference on Microelectronics, Computing and Communication Systems. Berlin, Germany: Springer, 2019: 91-107.

[ 8 ] HUANG P, HE X, GAO J, et al. Deep structured semantic model produced using click-through data[EB/OL]. [2019-11-10]. <https://patents.justia.com/patent/20150074027>.

[ 9 ] KISHORE A V. Word2vec[M]//AYYADEVARA V K. Pro machine learning algorithms. Berkeley, USA: Apress, 2018.

[ 10 ] PENNINGTON J, SOCHER R, MANNING C. Glove: global vectors for word representation[C]//Proceedings of 2014 Conference on Empirical Methods in Natural Language Processing. Stroudsburg, USA: Association for Computational Linguistics, 2014: 1532-1543.

[ 11 ] FENG Xingjie, ZHANG Le, ZENG Yunze. Question similarity calculation model based on multi-attention CNN[J]. Computer Engineering, 2019, 45(9): 284-290. (in Chinese)  
冯兴杰, 张乐, 曾云泽. 基于多注意力CNN的问题相似度计算模型[J]. 计算机工程, 2019, 45(9): 284-290.

[ 12 ] MUELLER J, THYAGARAJAN A. Siamese recurrent architectures for learning sentence similarity[C]//Proceedings of the 30th AAAI Conference on Artificial Intelligence. New York, USA: ACM Press, 2016: 2786-2792.

[ 13 ] MA Huifang, LIU Wen, LI Zhixin, et al. Combining coupled distance discrimination and strong classification features for short text similarity calculation[J]. Acta Electronica Sinica, 2019, 47(6): 1331-1336. (in Chinese)

- 马慧芳,刘文,李志欣,等. 融合耦合距离区分度和强类别特征的短文本相似度计算方法[J]. 电子学报, 2019,47(6):1331-1336.
- [14] DENG Han, ZHU Xinhua, LI Qi, et al. Sentence similarity calculation based on syntactic structure and modifier[J]. Computer Engineering, 2017, 43(9): 240-244, 249. (in Chinese)  
邓涵,朱新华,李奇,等. 基于句法结构与修饰词的句子相似度计算[J]. 计算机工程, 2017, 43(9): 240-244, 249.
- [15] YANG Meng, LI Peifeng, ZHU Qiaoming. Sentence similarity on structural representations[EB/OL]. [2019-11-10]. <http://tcci.ccf.org.cn/conference/2016/papers/112.pdf>.
- [16] ZHANG Xiaochuan, YU Linfeng, ZHANG Yihao. Multi-feature fusion for short text similarity calculation based on LDA[J]. Computer Science, 2018, 45(9): 273-277. (in Chinese)  
张小川,余林峰,张宜浩. 基于LDA的多特征融合的短文本相似度计算[J]. 计算机科学, 2018, 45(9): 273-277.
- [17] AFZAL Z, GARCIA J, LINDSKOG S, et al. Slice distance: an insert-only levenshtein distance with a focus on security applications[C]//Proceedings of the 9th IFIP International Conference on New Technologies, Mobility and Security. Washington D. C., USA: IEEE Press, 2018: 1-5.
- [18] BARD G V. Spelling-error tolerant, order-independent pass-phrases via the Damerau-Levenshtein string-edit distance metric[EB/OL]. [2019-11-10]. <https://www.semanticscholar.org/paper/Spelling-Error-Tolerant%2C-Order-Independent-via-the-Bard/03736bfb64f7c7cebb21a1696358c9541b96990c? p2df>.
- [19] GAO Xuyang. A question-answering system based on sentence similarity comparison[D]. Hangzhou: Zhejiang University, 2018. (in Chinese)  
高旭杨. 基于语句相似度计算的FAQ问答系统设计[D]. 杭州:浙江大学, 2018.
- [20] MA Yongqi, HAN Depei, MENG Lirong, et al. Lexical semantic similarity algorithm based on How-net[J]. Computer Engineering, 2018, 44(6): 151-155. (in Chinese)  
马永起,韩德培,蒙立荣,等. 基于How-net的词语语义相似度算法[J]. 计算机工程, 2018, 44(6): 151-155.
- [21] LIU Qinglei, GU Xiaofeng, LI Jianping. Researches of Chinese sentence similarity based on HowNet[C]//Proceedings of 2010 International Conference on Apperceiving Computing and Intelligence Analysis. Washington D. C., USA: IEEE Press, 2010: 26-29.
- [22] WU Jian, WU Zhaohui, LI Ying, et al. Web service discovery based on ontology and similarity of words[J]. Chinese Journal of Computers, 2005, 28(4): 595-602. (in Chinese)  
吴健,吴朝晖,李莹,等. 基于本体论和词汇语义相似度的Web服务发现[J]. 计算机学报, 2005, 28(4): 595-602.
- [23] HOCHREITER S, SCHMIDHUBER J. Long short-term memory[J]. Neural Computation, 1997, 9(8): 1735-1780.
- [24] WANG Guochun, ZHENG Shanhong, ZHAO Hui, et al. Calculation method of comprehensive ontology similarity based on stage progression[J]. Journal of Jilin University (Information Science Edition), 2014, 32(2): 92-95. (in Chinese)  
王国春,郑山红,赵辉,等. 基于阶段递进的综合本体相似度计算方法[J]. 吉林大学学报(信息科学版), 2014, 32(2): 92-95.
- [25] LI Xiao, XIE Hui, LI Lijie. Research on sentence semantic similarity calculation based on Word2vec[J]. Computer Science, 2017, 44(9): 256-260. (in Chinese)  
李晓,解辉,李立杰. 基于Word2vec的句子语义相似度计算研究[J]. 计算机科学, 2017, 44(9): 256-260.
- [26] ZHAI Yandong, WANG Kangping, ZHANG Dongna, et al. An algorithm for semantic similarity of short text based on WordNet[J]. Acta Electronica Sinica, 2012, 40(3): 617-620. (in Chinese)  
翟延冬,王康平,张东娜,等. 一种基于WordNet的短文本语义相似性算法[J]. 电子学报, 2012, 40(3): 617-620.

编辑 吴云芳