



## 基于改进句子相似度算法的释义识别研究

陈俊月, 郝文宁, 张紫萱, 唐新德, 康睿智, 莫 斐

(陆军工程大学 指挥信息系统学院, 南京 210000)

**摘 要:** 针对现有句子相似度算法无法处理同义词、准确率低和复杂度高等不足, 结合词向量技术改进 Levenshtein 相似度算法和 Jaccard 系数, 提出一种新的句子相似度算法用于释义识别, 并对多种句子相似度算法的优劣进行分析, 设计多相似度特征组合的应用模式。基于 MRPC 释义识别数据集的实验结果表明, 使用该算法的释义识别模型准确率与 F1 值分别达到 74.4% 和 83.1%, 与使用 TF-IDF 算法、词袋算法等传统算法的模型相比识别性能更优。  
**关键词:** 句子相似度; Jaccard 系数; Levenshtein 距离; 词向量; 释义识别; 多特征组合

开放科学(资源服务)标志码(OSID):



**中文引用格式:** 陈俊月, 郝文宁, 张紫萱, 等. 基于改进句子相似度算法的释义识别研究[J]. 计算机工程, 2020, 46(9): 76-82.

**英文引用格式:** CHEN Junyue, HAO Wenning, ZHANG Zixuan, et al. Research on paraphrase identification based on improved sentence similarity algorithm[J]. Computer Engineering, 2020, 46(9): 76-82.

## Research on Paraphrase Identification Based on Improved Sentence Similarity Algorithm

CHEN Junyue, HAO Wenning, ZHANG Zixuan, TANG Xinde, KANG Ruizhi, MO Fei  
(Institute of Command Information System, Army Engineering University, Nanjing 210000, China)

**[Abstract]** The existing sentence similarity algorithms fail to process synonyms and are faced with low accuracy and high complexity. To address the problems, this paper proposes a new sentence similarity algorithm for paraphrase identification by using the word embedding technique to improve the Levenshtein similarity algorithm and Jaccard index. Also, the advantages and disadvantages of the sentence similarity algorithms are briefly analyzed, and the application mode of multi-similarity feature combination is designed. Experimental results on MRPC paraphrase recognition data set show that the accuracy rate and F1 value of the paraphrase identification model using this algorithm are 74.4% and 83.1% respectively. Compared with the models using TF-IDF algorithm, Bag of Words (BoW) algorithm and other traditional algorithms, it has better recognition performance.

**[Key words]** sentence similarity; Jaccard index; Levenshtein Distance (LD); word embedding; paraphrase identification; multi-feature combination

**DOI:** 10.19678/j.issn.1000-3428.0055313

### 0 概述

随着计算机技术的飞速发展和互联网的全面普及, 全球数字化信息的存储量爆发式增长。丰富的信息资源既为人们的生活和生产带来了便利, 但同时也导致了信息重复冗余的问题。因此, 文本语义相似度研究应运而生, 其可作为承载语义信息的一种重要方式<sup>[1]</sup>。文本语义相似度计算具有广阔的实

际应用背景, 例如论文查重、搜索引擎去重等, 受到研究者的广泛关注<sup>[2-4]</sup>。

释义识别<sup>[5-6]</sup>用于判断任意两个句子或短语其语义是否相同, 可看作是文本语义相似度计算的子任务。目前针对句子级别的文本语义相似度即句子相似度有许多较为成熟的计算方法, 例如编辑距离 (Levenshtein Distance, LD) 算法<sup>[7]</sup>、Jaccard 系数<sup>[8]</sup>和词频-逆文档频率 (Term Frequency-Inverse Document

**基金项目:** 国家自然科学基金“效果驱动背景下基于层次任务网的作战行动序列快速生成及动态修复方法”(61806221)。

**作者简介:** 陈俊月 (1995—), 女, 硕士研究生, 主研方向为自然语言处理; 郝文宁 (通信作者), 教授; 张紫萱、唐新德, 硕士研究生; 康睿智, 博士研究生; 莫 斐, 硕士研究生。

**收稿日期:** 2019-06-27

**修回日期:** 2019-08-28

**E-mail:** hwnbox@163.com

Frequency, TF-IDF) 算法等,但是这些方法都有着明显的不容忽视的局限性和缺点:编辑距离算法仅考虑两个句子相互转换的操作次数,忽视了操作本身引起的语义差异;基于句子间共现词计算句子相似度的 Jaccard 系数只考虑句子的字面量,忽视了词语的语义信息<sup>[9]</sup>;TF-IDF 算法需要一个大规模语料支撑统计运算<sup>[6]</sup>,而对于某些领域(如军事领域),这样的大规模语料是难以获得的。近年来,随着深度学习技术的兴起,许多基于深度学习的文本相似度算法也随之涌现。但这些算法往往需要大量的语料作为支撑,同时由于深度神经网络本身的特点,因此其对计算时间和计算资源的要求都很高,而深度学习本身的黑箱性质也导致模型难以解释。

词向量技术在自然语言处理(Natural Language Processing, NLP)领域被广泛应用,其通过无监督学习从大规模语料中获取词的高维向量表示,能够很好地捕捉词语的语法和语义信息<sup>[9]</sup>并准确计算词语间的语义相似度。本文基于词向量对 Levenshtein 相似度算法和 Jaccard 系数进行改进,提出一种新的句子相似度算法。通过分析对比多种句子相似度算法的优缺点和应用场景,设计多相似度特征的组合应用模式,并使用改进后的句子相似度算法在 MRPC<sup>[10]</sup> 释义识别数据集上进行释义识别实验。

## 1 相关工作

句子相似度计算可分为基于字重叠的方法、基于语言学的方法和基于语料库的方法 3 类<sup>[6]</sup>。

基于字重叠的方法主要通过两个句子共有的词汇量来计算句子间的相似度,典型代表为 Jaccard 系数,其以两个句子中词汇交集与词汇并集的比值作为句子相似度。基于字重叠方法的基本思想是两个句子的构成短语或词汇的重叠个数越多,其相似度就越大,但这类方法都是字面层次的文本比较,未考虑词语本身含义,因而使用范围受到一定限制(如无法解决同义词问题)。

基于语言学的方法通过计算构成两个句子的词之间的语义关系和语法成分来确定句子间的相似度(如基于句法分析找到句子中各部分的依存关系或语义关系,在计算相似度的同时考虑词语相似度和关系相似度),该方法支持较为丰富的语义,但是句子本身的复杂性为框架分析带来较大的难度和工作量。

基于语料库的方法将文本表示为一系列词语的组合,把基于语料库统计的向量的余弦相似度作为句子对的相似度,而不考虑词语的语序和含义,典型代表为 TF-IDF 算法。TF-IDF 的基本思想是字词的重要性随其在文档中出现的次数成正比增加,但同时会随其在语料库中出现的频率反比下降。这类方法需要一个大的文本语料库做统计运算,而这类语料库在某些具体领域是难以获得的,例如军事领域。

随着词向量在自然语言学习领域的广泛使用,出现了大量基于词向量的句子表示学习方法,主要可分为无监督学习和有监督学习两类。

无监督学习方法包括 TF-IDF 加权句向量<sup>[11]</sup>、SIF 加权句向量<sup>[12]</sup>、Paragraph Vector DM/DBOM<sup>[13]</sup>、FastSent<sup>[14]</sup>、Skip-Thought Vector<sup>[15]</sup> 和 SDAE<sup>[14]</sup> 等。TF-IDF 加权句向量和 SIF 加权句向量是基于词袋(Bag of Words, BOW)模型的改进方法,这类方法通过对句子中词向量的加权组合得到句子的向量表示,忽略了词序信息,但简单高效,在一些任务中取得了较好的结果。Paragraph Vector DM/DBOM 分别基于 word2vec 中的 CBOW 模型和 skip-gram 模型<sup>[16]</sup>。Skip-Thought Vector 模型利用 word2vec 中的 skip-gram 模型进行从词级别到句子级别的推广,即对当前句子进行编码后再对其周围的句子进行预测。该模型采用 Encoder-Decoder 架构,由于使用了 GRU 网络,因此模型训练速度慢。SDAE(Sequential Denoising AutoEncoder)采用基于 LSTM 的 Encoder-Decoder 模型,包括编码器和解码器两部分,输入信息通过编码器产生编码信息,再通过解码器得到输出信息,模型的目标是使输出信息和输入信息越来越接近,相较于 Skip-Thought Vector 的优点是只需要输入单个句子,不要求句子所在的文本是有序的,而 Skip-Thought 的输入必须是 3 个有序的句子。FastSent 利用 Skip-Thought Vector 的设计思想以及 BOW 形式的编码方式,使得模型训练速度得到大幅提高。此外,研究者还提出一种 FastSent 的变体模型 FastSent + AE,其不仅预测前后两个句子中的词,同时还预测本身句子中的词。

有监督学习方法包括 InferSent<sup>[17]</sup> 和 GenSen<sup>[18]</sup> 等。InferSent 采用了迁移学习的方法,其从 SNIL 数据集<sup>[19]</sup> 训练得到 Encoder,然后使用预训练的 Encoder 生成句向量用于其他任务。该模型能够获得较优的结果,但其采用 BiLSTM 网络,训练速度较慢,而且对于数据集规模的要求较高。GenSen 则是一种基于多任务学习的模型,其同时在多个任务和多个数据源上进行训练但共享相同的 Sentence Embedding,这与 Skip-Thought Vector 基本一致,主要区别在于 Encoder 部分使用的是 Bi-GRU,因此,该模型训练速度同样较慢。

## 2 句子相似度算法

### 2.1 全匹配相似度

文献[20]提出了用于机器阅读理解文档-问题(QE)全匹配特征。对于文档中的某个词,假设该词在问题中出现,则其 QE 全匹配特征为 1,否则为 0。该特征从词的共现出发,一般而言,文档中包括问题词的部分更可能含有答案。本文基于这一思路,提出句子间的全匹配相似度指标,定义如下:

**定义 1** 给定句子 T、S 以及预训练的词向量文件 emb, 对于句子 T 中的某个词 t, 其与句子 T 的全匹配特征为:

$$F_{\text{feature}}(t, S) = \max_{s \in S} \cos(t_{\text{emb}}, s_{\text{emb}}) \quad (1)$$

其中, s 为句子 S 中的某个词, 下标 emb 表示取该词的词向量。对于句子 T, 该句与 S 的全匹配相似度如式(2)所示:

$$F_{\text{sim}}(T, S) = \frac{\sum_{t \in T} F_{\text{feature}}(t, S) + \sum_{s \in S} F_{\text{feature}}(s, T)}{\|T\| + \|S\|} \quad (2)$$

其中,  $\|\cdot\|$  表示句子包含的词数。

全匹配特征能够反映句子间词是否共现。一般而言, 若句子 T 中的词在句子 S 中出现, 那么这两个句子更可能相似。考虑到自然语言中存在大量的多词一义现象, 因此, 本文不直接考察文本词是否在查询中出现, 而是通过词向量之间的相似度度量某个文本词的词义是否在查询中出现。

## 2.2 Jaccard 系数及其改进

### 2.2.1 Jaccard 系数

Jaccard 系数又称为 Jaccard 相似系数, 可用于比较有限样本集之间的相似度与差异性, 系数值越大, 样本相似度越高<sup>[9]</sup>。直观上而言, 两个句子的相同部分越大, 共现词汇数目越多, 它们的相似度应该越高。因此, 可以使用 Jaccard 系数计算两个句子间的相似度, 如式(3)所示:

$$\text{Jaccard}(S, T) = \frac{\|\text{Inter}(S, T)\|}{\|\text{Union}(S, T)\|} \quad (3)$$

其中,  $\text{Inter}(S, T)$  表示句子 S 和句子 T 之间的词汇交集,  $\text{Union}(S, T)$  表示句子 S 和句子 T 之间的词汇并集。

### 2.2.2 改进的 Jaccard 系数

传统的 Jaccard 系数未考虑词的语义, 只是单纯地基于共现词计算句子的相似度, 无法处理多词一义的情况, 如 “I hava a computer.” 和 “I have a PC.”, 显然两句在语义上是相等的, 而传统 Jaccard 系数在计算过程中却无法识别 “computer” 和 “PC” 为语义一致, 所得到的 Jaccard 系数仅为 0.6。

词向量是每个词语在语义层面的高维向量表示, 本文结合词向量对传统的 Jaccard 系数进行改进。改进后的 Jaccard 系数定义如下:

**定义 2** 给定句子 T、S、预训练词向量文件 emb 以及词向量相似度阈值  $\alpha$  ( $\alpha \geq 0$ )。T 和 S 间的改进 Jaccard 系数如式(4)所示:

$$\text{Jaccard}'(T, S) = \frac{\sum_{t \in T} \sum_{s \in S} \cos(t_{\text{emb}}, s_{\text{emb}})}{\|T\| \cdot \|S\|} \quad (4)$$

其中, 下标 emb 代表取该词的词向量,  $\|T\| \cdot \|S\|$  表示句子 T 和 S 的长度乘积。

在改进的 Jaccard 系数定义式中, 分子部分代表

句子 T 和 S 中词向量的相似度之和, 相较于传统的 Jaccard 系数方法, 改进后的 Jaccard 词向量匹配特征由于引入了词向量, 不仅能够反映句子间词的共现情况, 而且还能反映词向量的共现值。分母部分实际上包括了分子部分以及句子 T 和 S 中词向量的不共现度, 因此,  $0 \leq \text{Jaccard}(T, S) \leq 1$ 。

## 2.3 Levenshtein 相似度算法

### 2.3.1 Levenshtein 距离及相似度算法

Levenshtein 距离是指两个字符串之间由原字符串 S 转换为新字符串 T 所需要的最少编辑次数, 允许的编辑操作包括插入、替换、删除<sup>[21]</sup>。

基于 Levenshtein 距离能够计算字符串间的相似度, 如式(5)所示:

$$L_{\text{sim}}(S, T) = 1 - \frac{l}{\max(S, T)} \quad (5)$$

其中, l 为字符串 S 与 T 间的 Levenshtein 距离。  $L_{\text{sim}}$  值越大, 表示两个字符串相似度越高;  $L_{\text{sim}}$  值越小, 表示两个字符串相似度越低。同样, 对于句子而言, 以词为单位进行插入、删除和替换操作即可得到句子级别的 Levenshtein 相似度。

### 2.3.2 基于编辑操作排序的路径回溯算法

目前对于 Levenshtein 相似度的求解一般使用动态规划算法。假设有两个字符串 S 和 T,  $S = \{s_1, s_2, \dots, s_N\}$ ,  $T = \{t_1, t_2, \dots, t_M\}$ , 建立 S 与 T 的  $(N+1)(M+1)$  阶 LD 矩阵, 如式(6)所示:

$$L_{(N+1)(M+1)}^D = \{d_{ij}\}, 0 \leq i \leq N+1, 0 \leq j \leq M+1 \quad (6)$$

在 LD 矩阵中, 第 1 列代表字符串 S, 第 1 行代表字符串 T,  $d_{ij}$  表示从字符串  $\{s_1, s_2, \dots, s_i\}$  到字符串  $\{t_1, t_2, \dots, t_j\}$  所需的最少编辑次数。矩阵填充公式如下:

$$d_{ij} = \begin{cases} i, & i=0 \\ j, & j=0 \\ \min(d_{i-1, j-1}, d_{i-1, j}, d_{i, j-1}) + a_{ij}, & i, j > 0 \end{cases}$$

$$a_{ij} = \begin{cases} 0, & s_i = t_j \\ 1, & s_i \neq t_j \end{cases}$$

其中,  $i = 1, 2, \dots, N, j = 1, 2, \dots, M$ , 矩阵右下角元素  $d_{NM}$  为字符串 S 和 T 之间的 Levenshtein 距离, 记为  $l_d$ 。

在 LD 矩阵基础上, 可以回溯得到编辑路径。需要注意的是, 在编辑路径回溯时可能产生多条编辑路径<sup>[16]</sup>, 而通过实验可以发现, 不同编辑路径计算得到的 Levenshtein 词向量距离是不同的。为避免回溯路径对相似度造成影响, 本文设计基于 LD 矩阵的编辑路径回溯算法, 通过对编辑操作的优先级排序得到唯一的回溯路径。算法描述如下:

**算法** 基于编辑操作排序的编辑路径回溯

1. Input S, T, operation\_rank

// 输入源字符串、目标字符串和编辑操作排序

```

2. Initialize LD matrix, i = len(S), j = len(T), list_operation = []
//初始化 LD 矩阵、回溯指针及回溯路径
3. While i >= 0 and j >= 0: //编辑路径回溯
{
coordinate = [i, j] //记录当前编辑操作发生的位置
If i == 0: operation = "insert";
elif j == 0: operation = "delete";
Else: //根据编辑距离和编辑操作排序选择编辑操作
{
candidate_operation = {"replace": LD(i-1, j-1), "insert":
LD(i, j-1), "delete": LD(i-1, j)}
sorted_operation = sort(candidate_operation, by = lambda value,
operation_rank: value, descending = False)
operation = sorted_operation[0]
} //更新回溯指针
If operation == "delete": j = j - 1;
elif operation == "insert": i = i - 1;
else: i = i - 1, j = j - 1;
list_operation.append([operation, coordinate]) //更新回
溯路径
}
4. Output list_operation //输出回溯路径

```

由于算法指定了编辑操作的优先级排序,因此在进行 LD 矩阵回溯时,每一步的回溯操作都是唯一的,即能够得到唯一的回溯路径。根据分析可知,该算法的空间复杂度和时间复杂度仅与 2 个字符串的长度有关,其中空间开销即 LD 矩阵的存储开销为  $O(MN)$ , 时间开销即回溯的次数,若  $M > N$ , 则时间复杂度为  $O(M)$ , 否则为  $O(N)$ 。需要注意的是,“replace”本质上并不是 Levenshtein 算法规定的替换操作,而是指在进行字符串转换时直接从原字符串复制字符到目标字符串中,但是为了统一符号,本文将将其看作一种特殊的待替换字符与替换字符执行相同的替换操作。

### 2.3.3 改进的 Levenshtein 相似度算法

传统的 Levenshtein 相似度算法从字符串互相转换的角度出发计算需要的最少编辑操作次数,能够从一定层面上反映句子间的相似性。但是仅考虑所需编辑操作次数而不考虑编辑操作本身引起的语义差异是不够的。假设有句子 S、T 和 Q, 分别为“I have a computer.” “I have a PC.” 和 “I have a TV.”, 根据传统的 Levenshtein 相似度算法, 3 个句子两两之间都只需要一次替换操作即可完成转换。实际上句子 S 与 T 的语义一致, 而 S 与 Q、T 与 Q 的语义差异更大, 这是因为不同的编辑操作对句子语义的改变是不同的, 将“computer”替换为“PC”并未对语义造成影响, 因为“computer”与“PC”是等价的, 而将其替换为“TV”则会引起较大的语义改变。

针对上述问题, 本文通过引入词向量对 Levenshtein 距离算法进行改进。改进算法的具体步骤如下:

**步骤 1** 给定句子 T 和 S,  $l = \max(\|T\|, \|S\|)$ , 计算两者间的 LD 矩阵。

**步骤 2** 通过 LD 矩阵回溯编辑路径并记录路径。

**步骤 3** 对于编辑路径中的删除和插入操作, 将其转换为特殊的替换操作, 即删除操作等价于将待删除字符替换为空字符, 插入操作等价于将空字符替换为待插入字符。

**步骤 4** 对于编辑路径中的所有操作, 通过词向量计算替换操作两个字符的相似度并累加, 将得到结果记为  $L_{emb}$ , 即 Levenshtein 词向量距离。

**步骤 5** 输出改进的 Levenshtein 相似度  $1 - L_{emb}/l$ , 记为  $L_{sim'}$ 。

根据改进的 Levenshtein 相似度算法对例句 S、T 和 Q 计算, 从 S 转换到 T 和从 S 转换到 Q 都只需要一次替换操作, 但由于“computer”与“PC”的语义相近, “computer”与“TV”的语义相差较大, 因此有:  $L_{emb}(S, T) < L_{emb}(S, Q)$ , 最终可得:  $L_{sim'}(S, T) > L_{sim'}(S, Q)$ 。

可以看出, 改进后的 Levenshtein 相似度算法相较于传统 Levenshtein 相似度算法能够更准确地刻画句子之间的语义相关性。

## 2.4 多相似度特征组合分析

Jaccard 系数和 Levenshtein 相似度是目前常用的句子相似度计算方法, 两者从不同的角度刻画了句子间的相似度。Jaccard 系数从集合论的思想出发, 利用句子间共现词占全部词的比例反映句子的相似度, 但忽视了句子语序的影响, 而 Levenshtein 相似度则通过句子间相互转换所需操作次数来衡量句子相似度, 一定程度上考虑了句子间的语序。两种方法本质上都是基于字面量的匹配方法, 虽然取得了一定的效果, 但都无法解决多词一义问题。因此, 本文引入词向量这一词语语义的高维表示方法对上述算法进行改进, 提出一种新的相似度算法。改进后的 Jaccard 系数和 Levenshtein 相似度不仅能够反映句子间字面量的相似性, 而且能够应对多词一义的情况。然而需要指出的是, 改进后的 Jaccard 系数和 Levenshtein 相似度由于使用了词向量作为支撑, 因此词向量的质量对于算法性能有较大影响, 而对于一些专有名词如人名、地名或专业领域(如医学、生物、建筑等)词汇, 这些词汇往往不可替代且出现次数较少。因此, 词向量无法完全反映其语义信息甚至难以获得这类词的词向量, 而传统 Jaccard 系数和 Levenshtein 相似度在面对此类句子时则较为有效。

多种句子相似度指标的组合能够有效结合各类指标的优点。传统的 Jaccard 系数和 Levenshtein 相似度对于专有名词表现更好, 而改进的 Jaccard 系数和 Levenshtein 相似度则对通用词汇间的多词一义现象更具优势。全匹配特征类似于 Jaccard 系数, 其直接通过词的共现判断句子相似度, 但没有采用集合论方法, 因此能够反映词频信息。本文将句子对的

Jaccard 系数、Levenshtein 相似度、全匹配相似度、改进后的 Jaccard 系数与 Levenshtein 相似度共 5 种相似度指标相结合作为特征向量,共同描述句子对间的相似度关系,然后根据这一特征向量判断句子间的语义等价关系,从而避免信息的遗漏和丢失,增强算法的健壮性。

### 3 实验

#### 3.1 数据集与评价指标

本文选用微软发布的通用语料集 MRPC 进行实验。MRPC 数据集是一个用于释义识别任务的数据集,包含取自互联网新闻文章的 4 076 对训练句子和 1 725 对测试句子,每个句子对由两位评估人员判断其语义是否等价,该数据集样本分布如表 1 所示。

表 1 MRPC 数据集样本分布

Table 1 Sample distribution of MRPC dataset

| 样本类别  | 总数    | 训练集   | 测试集   |
|-------|-------|-------|-------|
| 总样本   | 5 801 | 4 076 | 1 725 |
| 等价样本  | 3 900 | 2 753 | 1 147 |
| 不等价样本 | 1 901 | 1 323 | 578   |

给出如下数据集示例,其中,“#1 String”和“#2 String”是被测试的句子对,Quality 表示句子对是否语义等价,取值为{0,1}。

Quality#1 ID#2 ID#1 String#2 String

1702876702977 Amrozi accused his brother, whom he called "the witness", of deliberately distorting his evidence. Referring to him as only "the witness", Amrozi accused his brother of deliberately distorting his evidence.

021087052108831 Yucaipa owned Dominick's before selling the chain to Safeway in 1998 for \$2.5 billion. Yucaipa bought Dominick's in 1995 for \$693 million and sold it to Safeway for \$1.8 billion in 1998.

MRPC 数据集以准确率(accuracy)和 F1 值作为评价指标,其定义如下:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

其中:TP 为预测为等价并且实际上等价的样本数量;TN 为预测为不等价并且实际上不等价的样本数量;FP 为预测为等价但实际上不等价的样本数量;FN 为预测为不等价但实际上等价的样本数量<sup>[6]</sup>。

#### 3.2 相似度计算与模型训练

首先对句子进行预处理,然后计算句子对的 Jaccard 系数、Levenshtein 相似度、全匹配相似度、改

进后的 Jaccard 系数与 Levenshtein 相似度 5 种相似度指标。其中,分词方法选用 python 的 spacy 包,词向量选用 300 维的谷歌预训练词向量<sup>[22]</sup>(下载地址为 [nlp.stanford.edu/data/glove.840B.300d.zip](http://nlp.stanford.edu/data/glove.840B.300d.zip)),改进 Jaccard 系数算法的参数,并将 Levenshtein 距离计算中路径回溯算法的编辑操作优先级排序为[replace,delete,insert]。

由于各类相似度取值都较为接近,简单的特征加权算法会丢失大量信息,因此在完成各类相似度计算后,采用分类模型进行释义识别。以 5 种相似度指标值为样本特征,以句子对是否语义等价为类别标签,选用多种常用机器学习分类算法对训练数据进行训练,包括梯度提升树(Gradient Boosting Decision Tree,GBDT)、支持向量机(Support Vector Machine,SVM)、朴素贝叶斯(Naïve Bayes,NB)、K 近邻分类(K-Nearest Neighbor algorithm,KNN)、决策树(Decision Tree,DT)、随机森林(Random Forest,RF)和逻辑回归分类(Logistic Regression,LR)。

#### 3.3 实验结果与分析

##### 3.3.1 分类算法对比实验

表 2 记录了基于多相似度特征不同分类算法在 MRPC 数据集上的表现,其中黑体表示最优结果。可以看出,表现最好的是 SVM 算法和 LR 算法。SVM 算法的 F1 值达到 83.69%,准确率为 73.62%。LR 算法虽然在 F1 值上略逊色于 SVM 算法,为 82.12%,但准确率的表现更好,达到了 74.14%。KNN 算法、DT 算法和 RF 算法整体表现较差,笔者认为这与数据中类别不平衡和特征相关性有关。由表 1 可见:MRPC 数据存在一定的类别不平衡现象,其语义等价样本数和语义不等价样本数的比值约为 2:1,同时作为样本特征的 5 类相似度指标存在一定的相关性;KNN 算法容易受到类别不平衡影响,决策树算法不仅受类别不平衡影响较大,而且还容易过拟合,因此,两者表现较差,而基于决策树的随机森林算法同样无法避免这个问题;SVM 算法基于结构风险最小化原则,能够避免过拟合问题,泛化能力强,并且 SVM 是一个凸优化问题,其局部最优解必定为全局最优解,而 LR 算法不容易受到特征间相关性的影响,因此,这两种算法取得了较好的实验结果。

表 2 不同分类算法在 MRPC 数据集上的准确率和 F1 值

Table 2 Accuracies and F1 values of different classification algorithms on MRPC dataset

| 分类算法    | 准确率          | F1 值         |
|---------|--------------|--------------|
| SVM 算法  | 73.92        | <b>82.69</b> |
| LR 算法   | <b>74.14</b> | 82.12        |
| GBDT 算法 | 72.41        | 80.91        |
| NB 算法   | 72.23        | 79.61        |
| KNN 算法  | 68.06        | 77.33        |
| RF 算法   | 67.71        | 76.13        |
| DT 算法   | 63.48        | 72.30        |

由于表2所示的实验结果是在未做进一步模型调参下获得的,即使是表现最差的决策树算法也依然取得了准确率63.48%和F1值72.30%的结果,表明输入特征能够有效区分句子间的语义相似度。

### 3.3.2 本文算法与其他算法的对比实验

根据表2不同分类算法的表现结果,本文选择SVM算法训练分类模型,同时应用超参数搜索和交叉检验等方法提升模型表现,其与现有其他模型的准确率和F1值比较如表3所示,其中黑体表示最优结果,†表示在文献[20]基础上通过迁移学习得到的新模型。可以看出,本文模型准确率为74.4%,F1值为82.6%,位居第三,并且与前两名实验结果差距较小。

表3 使用不同句子相似度算法的释义识别模型性能对比  
Table 3 Performance comparison of paraphrase recognition models using different sentence similarity algorithms %

| 实验组号 | 释义识别模型                   | 准确率         | F1 值        |
|------|--------------------------|-------------|-------------|
| 1    | Unigram-TFIDF            | 73.6        | 81.7        |
|      | Paragraph Vec (DBOW)     | 72.9        | 81.1        |
|      | SDAE                     | 73.7        | 80.7        |
|      | Word2vec BOW†            | 72.5        | 81.4        |
|      | fastText BOW†            | <b>73.9</b> | <b>82.0</b> |
|      | Glove BOW†               | 72.1        | 80.9        |
|      | Glove Position Encoding† | 72.5        | 81.2        |
| 2    | FastSent                 | 72.2        | 80.3        |
|      | Skip Thought             | <b>73.0</b> | <b>82.0</b> |
| 3    | InferSent (on SST)       | 72.7        | 80.9        |
|      | InferSent (on SNLI)      | 75.1        | 82.2        |
|      | InferSent (on AllNLI)    | 76.2        | 83.1        |
|      | GenSen†                  | <b>78.6</b> | <b>84.4</b> |
|      | 本文模型                     | 74.4        | 82.6        |

表3中的模型都需要大规模语料库或深度学习模型进行支撑。如第1组和第2组中的Unigram-TFIDF、Paragraph Vec和SDAE、FastSent、Skip Thought等无监督表示学习方法都是在Toronto book corpus(该语料库大小约70 MB,由来自7 000多本书的有序句子组成)上训练得到,并且这两组模型最好的实验结果分别为准确率73.9%、F1值82.0%和准确率73.0%、F1值82.0%,相较本文模型的实验结果仍有一定差距。第3组中的方法通过有监督训练学习得到句子表示,然后基于句子的表示进行释义识别,仅有InferSent和GenSen两种模型比本文模型表现得更好。但是InferSent模型的结果是基于SNLI和MultiGenre NLI(该数据集是SNLI的增强版本,包括433k个句子对)两个数据集<sup>[23]</sup>训练得到的,而当InferSent模型在SST数据集<sup>[24]</sup>(该数据集仅包括

12k个样本)上进行训练时仅能得到准确率72.7%和F1值80.9%的实验结果,相较于本文模型的实验结果低了2个百分点,说明该模型的表现受到语料数量的限制。此外,InferSent模型使用了BiLSTM-Maxpooling作为编码器,这导致其训练时长和计算资源需求都十分巨大,根据官方代码,该模型的参数量约为47 MB。GenSen模型基于GRU的Encoder-Decoder架构,同时使用了多任务学习,这种方法同样对于计算资源和语料的要求都十分巨大,而在不进行多任务学习的情况下其表现为准确率72.4%、F1值81.6%,与本文模型相比仍略有差距。

相比上述模型,本文基于改进文本相似度的释义识别模型仅使用了通用的预训练词向量,计算过程也只包括相似度指标计算和简单的机器学习分类模型训练两部分,因而实验结果较优,证明了本文对文本相似度算法的改进是有效的,并且可将该算法作为一种简单但优秀的基线模型用于后续的释义识别任务研究。

## 4 结束语

本文针对现有句子相似度算法存在的不足,结合词向量技术对传统的Levenshtein相似度算法和Jaccard系数进行改进,提出一种新的句子相似度算法。在通用MRPC数据集上的实验结果表明,该算法能够提高准确率和F1值,可作为一种简单有效的基线模型用于后续的释义识别任务。下一步将拓展更多的句子相似度度量指标,同时加快算法的计算速度。

## 参考文献

- [1] WU Shaohong, PENG Dunlu, YUAN Weiwei, et al. MGSC: a multi-granularity semantic cross model for matching short texts [J]. Journal of Chinese Computer Systems, 2019, 40(6): 1148-1152. (in Chinese)  
吴少洪,彭敦陆,苑威威,等. MGSC:一种多粒度语义交叉的短文本语义匹配模型[J]. 小型微型计算机系统, 2019, 40(6): 1148-1152.
- [2] JIN Bo, SHI Yanjun, TENG Hongfei. Similarity algorithm of text based on semantic understanding [J]. Journal of Dalian University of Technology, 2005, 45(2): 291-297. (in Chinese)  
金博,史彦军,滕弘飞. 基于语义理解的文本相似度算法[J]. 大连理工大学学报, 2005, 45(2): 291-297.
- [3] ZHAO Zhen, WU Ning, SONG Panpan. Sentence semantic similarity calculation based on multi-feature fusion [J]. Computer Engineering, 2012, 38(1): 171-173. (in Chinese)  
赵臻,吴宁,宋盼盼. 基于多特征融合的句子语义相似度计算[J]. 计算机工程, 2012, 38(1): 171-173.



- [4] LIU Hongzhe. Research on text semantic similarity calculation method [D]. Beijing: Beijing Jiaotong University, 2012. (in Chinese)  
刘宏哲. 文本语义相似度计算方法研究 [D]. 北京: 北京交通大学, 2012.
- [5] HUANG Jiangping, JI Donghong. Paraphrase identification based on sentence semantic distances [J]. Journal of Sichuan University (Engineering Science Edition), 2016, 48(6): 202-207. (in Chinese)  
黄江平, 姬东鸿. 基于句子语义距离的释义识别研究 [J]. 四川大学学报 (工程科学版), 2016, 48(6): 202-207.
- [6] KOZAREVA Z, MONTOYO A. Paraphrase identification on the basis of supervised machine learning techniques [C]// Proceedings of International Conference on Natural Language Processing (in Finland). Berlin, Germany: Springer, 2006: 524-533.
- [7] LI Yujian, LIU BO. A normalized Levenshtein distance metric [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007, 29(6): 1091-1095.
- [8] REAL R, VARGAS J M. The probabilistic basis of Jaccard's index of similarity [J]. Systematic Biology, 1996, 45(3): 380-385.
- [9] CHEN Zhenrui, DING Zhiming. Improved word representation based on GloVe model [J]. Computer Systems & Applications, 2019, 28(1): 194-199. (in Chinese)  
陈珍锐, 丁治明. 基于 GloVe 模型的词向量改进方法 [J]. 计算机系统应用, 2019, 28(1): 194-199.
- [10] DOLAN B. Unsupervised construction of large paraphrase corpora: exploiting massively parallel news sources [C]// Proceedings of the 20th International Conference on Computational Linguistics. San Diego, USA: Association for Computational Linguistics, 2004: 350-357.
- [11] WU H C, LUK R W P, WONG K F, et al. Interpreting TF-IDF term weights as making relevance decisions [J]. ACM Transactions on Information Systems, 2008, 26(3): 1-27.
- [12] ARORA S, LIANG Y, MA T. A simple but tough-to-beat baseline for sentence embeddings [C]// Proceedings of International Conference on Learning Representations. Toulon, France: [s. n.], 2017: 1-16.
- [13] HILL F, CHO K, KORHONEN A. Learning distributed representations of sentences from unlabeled data [C]// Proceedings of NAACL-HLT 2016. San Diego, USA: Association for Computational Linguistics, 2016: 1367-1377.
- [14] LOGESWARAN L, LEE H. An efficient framework for learning sentence representations [EB/OL]. [2019-05-10]. <https://arxiv.org/pdf/1803.02893.pdf>.
- [15] KIROS R, ZHU Y, SALAKHUTDINOV R, et al. Skip-thought vectors [C]// Proceedings of International Conference on Neural Information Processing Systems. Montreal, Canada: [s. n.], 2015: 1-11.
- [16] JIANG Hua, HAN Anqi, WANG Meijia, et al. Solution algorithm of string similarity based on improved Levenshtein distance [J]. Computer Engineering, 2014, 40(1): 222-227. (in Chinese)  
姜华, 韩安琪, 王美佳, 等. 基于改进编辑距离的字符串相似度求解算法 [J]. 计算机工程, 2014, 40(1): 222-227.
- [17] CONNEAU A, KIELA D, SCHWENK H, et al. Supervised learning of universal sentence representations from natural language inference data [EB/OL]. [2019-05-10]. <https://arxiv.org/pdf/1705.02364.pdf>.
- [18] SUBRAMANIAN S, TRISCHLER A, BENGIO Y, et al. Learning general purpose distributed sentence representations via large scale multi-task learning [C]// Proceedings of International Conference on Learning Representations. Vancouver, Canada: [s. n.], 2018: 1-16.
- [19] GONG Yichen, LUO Heng, ZHANG Jian. Natural language inference over interaction space [EB/OL]. [2019-05-10]. <https://arxiv.org/pdf/1709.04348.pdf>.
- [20] SU Jianlin. Q&A model based on CNN: DGCNN [EB/OL]. [2019-05-10]. <https://spaces.ac.cn/archives/5409>. (in Chinese)  
苏剑林. 基于 CNN 的阅读理解问答模型: DGCNN [EB/OL]. [2019-05-10]. <https://spaces.ac.cn/archives/5409>.
- [21] ZANG Runqiang, SUN Hongguang, YANG Fengqin, et al. Text similarity calculation method based on Levenshtein and TFRSF [J]. Computer and Modernization, 2018(4): 84-89. (in Chinese)  
藏润强, 孙红光, 杨凤芹, 等. 基于 Levenshtein 和 TFRSF 的文本相似度计算方法 [J]. 计算机与现代化, 2018(4): 84-89.
- [22] PENNINGTON J, SOCHER R, MANNING C. GloVe: global vectors for word representation [C]// Proceedings of Conference on Empirical Methods in Natural Language Processing. San Diego, USA: Association for Computational Linguistics, 2014: 1532-1543.
- [23] WILLIAMS A, NANGIA N, BOWMAN S R. A broad-coverage challenge corpus for sentence understanding through inference [EB/OL]. [2019-05-10]. <https://arxiv.org/pdf/1704.05426v4.pdf>.
- [24] SOCHER R, PERELYGIN A, WU J Y, et al. Recursive deep models for semantic compositionality over a sentiment treebank [EB/OL]. [2019-05-10]. <http://nlp.stanford.edu/sentiment/index.html>.