Software Engineering and Applications 软件工程与应用, 2022, 11(6), 1310-1318
Published Online December 2022 in Hans. http://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2022.116134

Hans汉斯

# 基于DFA的中拼混合敏感词过滤算法

**杨　扬，游福成**

北京印刷学院信息工程学院，北京

收稿日期：2022年11月5日；录用日期：2022年12月6日；发布日期：2022年12月16日

## 摘　要

**本文针对当前网络上通过各种干扰形式“伪装”的敏感词，提出了一种基于DFA的中拼混合敏感词过滤算法，解决了一般的系统过滤方法难以成功检测过滤该类敏感词的问题，提高了包含该类敏感词文本过滤的查全率和查准率。本文提出的算法包括中文拼音敏感词库的扩充算法、敏感词树的构建算法、待检测文本的预处理算法以及敏感词过滤算法，通过实验得到该算法查准率为100%，查全率约为95%~100%，算法复杂度较低，满足实际应用需要。**

## 关键词

**DFA，词库构建，敏感词过滤**

# Chinese Characters and Pinyin Mixed Sensitive Word Filtering Algorithm Based on DFA

**Yang Yang, Fucheng You**

School of Information Engineering, Beijing Institute of Graphic Communication, Beijing

Received: Nov. 5th, 2022; accepted: Dec. 6th, 2022; published: Dec. 16th, 2022

## Abstract

**Aiming at the sensitive words that are “camouflaged” through various interference forms on the current network, this paper proposes a Chinese character and Pinyin mixed sensitive word filtering algorithm based on DFA, which solves the problem that the general system filtering methods are difficult to successfully detect and filter such sensitive words, and improves the recall and precision of text filtering containing such sensitive words. The algorithm proposed in this paper includes the expansion algorithm of the Chinese characters and Pinyin mixed sensitive word li-**

文章引用: 杨扬, 游福成. 基于 DFA 的中拼混合敏感词过滤算法[J]. 软件工程与应用, 2022, 11(6): 1310-1318.
DOI: 10.12677/sea.2022.116134

**brary, the construction algorithm of the sensitive word tree, the pretreatment algorithm of the text to be detected, and the sensitive word filtering algorithm. Through the experiment, the precision of the algorithm is 100%, and the recall is about 95%~100%. The algorithm complexity is low so this algorithm meets the practical application needs.**

## Keywords

**DFA, Construction of Thesaurus, Sensitive Word Filtering**



## 1. 引言

如今的互联网已经成为人们工作生活中必不可少的一部分，它能够提供各种类型的服务和信息，但同时互联网的包容和开放也给各种反动、暴力、色情、虚假广告等不良信息提供了良好的生存传播条件，这些不良信息危害着网络环境、人们，尤其是青少年的身心健康以及社会的稳定发展。尤其随着经济的发展，移动设备的使用人群扩大且趋于低龄化，微信、QQ、微博等线上交流的方式更受青少年欢迎，也使得他们更容易受到网络不良信息的影响，因此对这些不良信息的过滤成为了重要的研究内容。

然而，与英文文本相比，中文文本中的字词之间并没有类似空格这样明显的分隔符号，且汉字数量庞大，组成的词语数量更是难以计数，因此中文文本的敏感词过滤较为困难。同时为了避免系统过滤屏蔽，中文敏感词的传播方式也日渐丰富，由原始形式直接传播，到现在的混合无关干扰字符传播、中文拼音混合传播等等，也给中文敏感词过滤增加了不小的难度。

传统的敏感词过滤算法较为简单，主要是单模式匹配和多模式匹配[1] [2]，后来有学者陆续对传统算法进行了改进，提出了 ST-DFA [3]等算法，但复杂度较高。刘邦国等人提出了一种面向 PDF 文本内容审查的高效多模式匹配算法[4]，但只适用于敏感词长度和数量都较大的情况；陈永杰等人提出了一种基于 Aho-Corasick 算法改进的多模式匹配算法[5]，但只适用于检测过滤原始形式的敏感词。因此本文提出了一种基于确定有限自动机(deterministic finite automaton, DFA)的中拼混合敏感词过滤算法，能够适用于敏感词长度数量不确定的情况且有效地过滤原始形式敏感词和干扰敏感词。

## 2. 敏感词过滤相关

### 2.1. 存在的问题

1) 干扰过滤。为了使包含敏感词的信息不被系统过滤得以传播，发布者通常会对敏感词进行“伪装”，常见的方法有添加无关字符干扰过滤，例如敏感词“金融危机”，可变形为“金@#融￥%危&……机”；使用中文拼音混合的形式干扰过滤，如变形为“jin 融 wei 机”；此外还有同音字替换如“金融尾鸡”，中文拆分如“金融危木几”等干扰方法甚至多种干扰方法混合使用。这些经过干扰方法“伪装”的敏感词大多能在人们阅读时被识别理解，但一般的系统过滤方法难以成功检测过滤。

2) 准确率。出现的敏感词能不能被准确的检测过滤，非敏感词会不会被错误的判断过滤。

### 2.2. 评价指标

对于敏感词过滤算法的评价指标主要有查全率、准确率和综合评价指标值[6]。

1)查全率 $R$，其中 $T$ 为文本包含的敏感词个数， $N$ 为算法正确过滤的敏感词个数。

$$R=\frac{N}{T}\times 100\% \tag{1}$$

2) 准确率 $P$，其中 $N$ 为算法正确过滤的敏感词个数， $M$ 为算法过滤的敏感词个数。

$$P=\frac{N}{M}\times 100\% \tag{2}$$

3) 综合评价指标值 $F$。

$$F=\frac{2\times R\times P}{R+P} \tag{3}$$

4) 算法复杂度，即时间复杂度和空间复杂度。

## 3. 算法设计

### 3.1. 算法总体设计

本文提出的算法的整体设计如图 1。

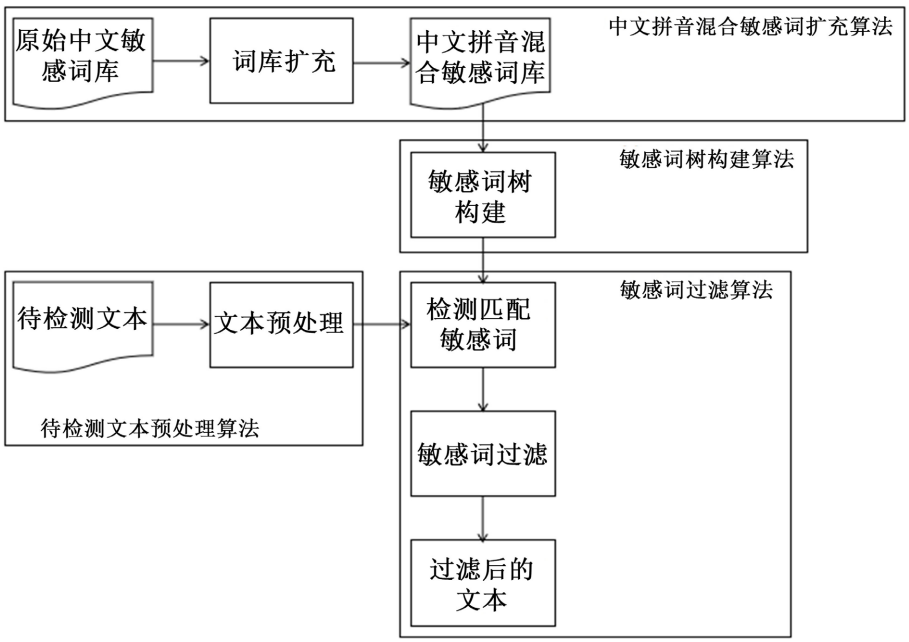


**Figure 1.** Overall process of sensitive word filtering algorithm
**图 1.** 敏感词过滤算法整体流程

### 3.2. 确定有限自动机

DFA 即确定有限自动机[7]，其数学模型为一个五元组 $M=(Q,\Sigma,\delta,q_0,F)$，其中，$Q$ 是一组有限状态的集合， $\forall q\in Q$， $q$ 称为 $M$ 的一个状态； $\Sigma$ 是一组输入符号的集合，称为字符表； $\delta$ 是状态转移函数，$\delta:Q\times\Sigma\to Q$，在一个状态 $q\in Q$ 读取到字符 $w\in\Sigma$，跳转到另一个状态 $p\in Q$，即 $(q,p,w)\in\delta$； $q_0$ 是起始状态； $F\subseteq Q$ 表示一组接受状态的集合。

在 DFA 中，状态和引起状态转换的事件都是确定的，状态和事件的数量是有限的[8]。如图 2 所示，状态 $S$ 读入字符 $a$ 转换为状态 $U$，即 $S\to a\to U$。

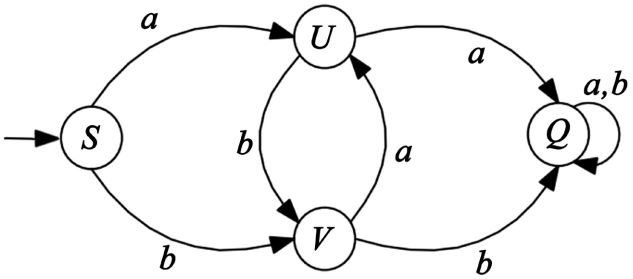


**Figure 2.** State transition
**图 2.** 状态转移

### 3.3. 中文拼音混合敏感词库扩充算法

为了避免中拼混合敏感词这一人工干扰过滤的情况，该算法利用排列组合的方式生成每个中文敏感词对应的所有中拼混合敏感词，如图 3 及公式(4)所示，一个中文敏感词由 $n$ 个字符组成，每个字符有汉字和拼音两种形式，因此一个中文敏感词对应的中拼混合敏感词一共有 $2^n$ 个。

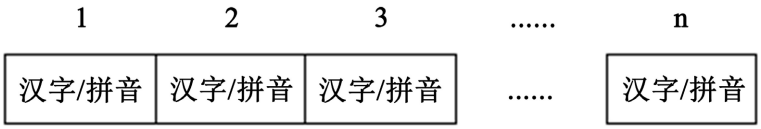


**Figure 3.** Principle of permutation and combination
**图 3.** 排列组合原理

$$2\times 2\times 2\times \cdots \times 2 = 2^n \tag{4}$$

该中文拼音混合敏感词库扩充算法具体如下：

1) 读取原始中文敏感词，以字为单位存入汉字列表 line；

2) 使用 get_pinyin()获取每个字的对应拼音并以整个拼音为单位存入拼音列表 pinyin_list，获取列表长度 length；

3) 新建中拼混合列表 zhongpin_list 用于存放生成的中拼混合敏感词；

4) 使用 random.sample()随机选择中拼混合敏感词的第一个字符形式并记录该字符；

5) 使用 random.sample()随机选择下一个字符形式并记录该字符，同时在两个字符中间添加空格字符作为分隔符；

6) 重复步骤 5)直到生成一个完整的中拼混合敏感词；

7) 使用__contains__()判断生成的中拼混合敏感词是否已经存在于中拼混合列表 zhongpin_list，不存在则添加进列表，存在则执行步骤 4)；

8) 当中拼混合列表 zhongpin_list 的长度等于 $2^n$ 时，一个中文敏感词扩充完成，将列表元素写入文件，继续执行步骤 1)直到所有中文敏感词扩充完成，得到中文拼音混合敏感词库。

### 3.4. 敏感词树构建算法

本文采用字典树结构构建敏感词树。

字典树[9]又称为 Trie 树、前缀树，具有以下特点：1) 树的根节点不存储数据，其他节点存储一个字符；2) 从根节点到叶节点的路径上的字符连接起来组成一个词。因此前缀相同的词语使用同一个分支，搜索时从词语的第一个字符开始查找对应的分支，能够显著地减少字符串比较，降低算法复杂度。

以敏感词“金融”和“金融危机”为例构建敏感词树如图 4 所示，前缀“金融”相同故使用同一分支，对应拼音整体存入一个叶节点，着色节点标志敏感词结束。

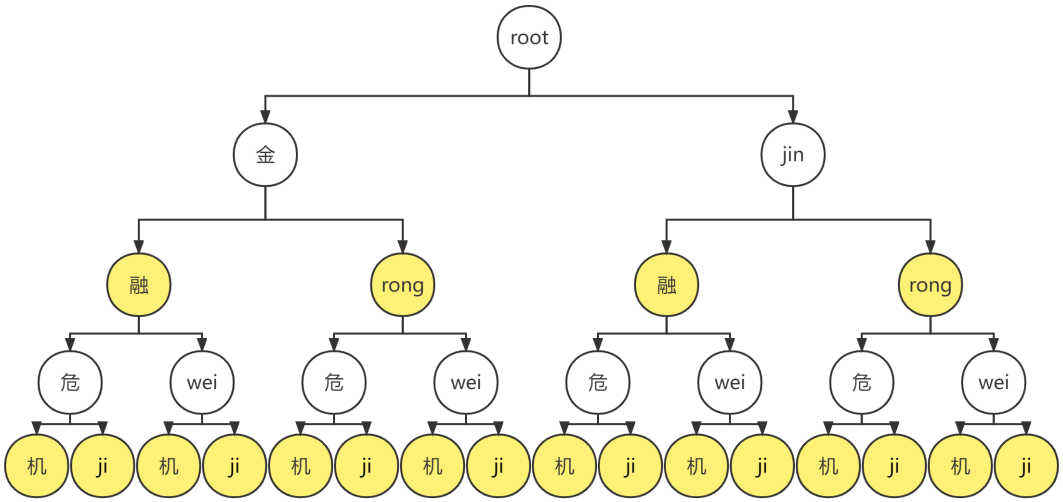


**Figure 4.** Sensitive word tree structure
**图 4.** 敏感词树结构

敏感词树构建具体算法如下：

1) 读取敏感词，使用 split()以空格为分隔符分割敏感词字符串，获取字符串列表；

2) 获取列表长度 length，定义 now_node 指向当前节点，初始值为根节点 root。定义 i 表示列表的第 i 个元素 $(0 \le i \le \text{length})$，初始值为 0；

3) 判断列表元素 word[i] 是否存在于 now_node.keys() 中，存在则进行赋值 now_node = now_node.get(word[i])，继续执行步骤 3)；不存在则新建节点 new_node，判断该节点是否为敏感词结束节点，是则进行标记，同时进行赋值 now_node[word[i]] = new_node，now_node = new_node；

4) 重复步骤 3)直至 i = length − 1，完成一个敏感词的添加；

5) 继续执行步骤 1)直至整个敏感词库添加完成，完成敏感词树的构建。

## 3.5. 待检测文本预处理算法

待检测文本中的拼音部分往往是连在一起没有明显分割的，这种情况不会影响人们阅读理解但是对于系统识别会产生影响导致无法准确过滤，因此需要对待检测文本进行处理分割出单个拼音。

单个拼音由声母和韵母组成，本文采用将声母大写的方法，根据大写字母分割出单个拼音。

同时识别并删除待检测文本中的无关干扰字符以免影响后续检测过滤。

具体算法如下：

1) 使用正则表达式去除掉除汉字、字母和数字外的字符；

2) 建立声母列表 smlist，判断待检测文本中是否包含列表元素，有则使用 upper()替换为大写；

3) 建立 nosm 列表，判断步骤 2)中是否将韵母中的 r，n，g 错误大写，有则使用 lower()替换回小写；

4) 建立 rep 字典，判断步骤 2)中是否将翘舌音声母中的 h 错误大写，有则替换回小写；

5) r，n，g 既可以作为韵母的结尾，也可以作为声母，故需要判断后一个字母是否为 smlist 列表中的元素，是则表示此处的 r，n，g 为韵母的结尾，保持小写，否则表示此处的 r，n，g 为声母，替换为大写；

6) 使用正则表达式以空格分隔汉字和拼音；

7) 使用 split()以空格为分割符分割字符串得到与原文本顺序一致、以单个汉字及单个拼音为元素的 detected 列表。

后续的敏感词检测过滤操作使用处理后的 detected 列表，可以直接将列表元素与敏感词树的节点进行匹配，提高匹配的准确性，匹配拼音节点时还可以提高匹配速度，减少匹配时间。

### 3.6. 敏感词过滤算法

使用上述算法完成词树构建和文本的预处理，根据敏感词树的结构，对处理后的文本进行敏感词检测过滤，将检测到的敏感词用符号“*”进行替换，检测算法基于 DFA 匹配模式，采用最小匹配规则。

具体算法如下：

1) 定义 matched_word_list 列表存放匹配到的完整的敏感词；now_map 表示构建的敏感词树；begin_index 表示检测的起始位置，初始值为 0；flag 表示是否检测到完整的敏感词，初始值为 False；tmp_flag 表示 detected 列表中匹配到的元素个数，初始值为 0；length 表示 detected 列表长度；i 表示当前 detected 列表元素位置$(0 \le i \le length)$。从 detected 列表的第一个元素开始执行以下操作，直到最后一个元素执行完成；

2) 获取 detected 列表的第 i 个元素 detected[i]，now_map.get(detected[i].lower())判断敏感词树上是否存在该节点，不存在则 tmp_flag = 0，begin_index = begin_index + 1，继续执行步骤 2)，存在则 now_map = now_map.get(detected[i])指向该节点，tmp_flag = tmp_flag + 1，判断该节点是否为敏感词结束节点，否则继续执行步骤 2)，是则 flag = True，将 detected 列表从 detected[begin_index]开始切片，长度为 tmp_flag，使用 join()转换成字符串格式，添加进列表 matched_word_list，begin_index = begin_index + tmp_flag；

3) 重复上述操作直至检测完成，得到匹配到的敏感词列表 matched_word_list，列表中元素的顺序与文本中敏感词的顺序一致；

4) 根据 matched_word_list 列表，将去除掉无关字符的文本中的敏感词使用 replace(num = 1)替换为相同长度的“*”字符串，得到过滤后的文本 result_txt。

## 4. 实验及结果分析

### 4.1. 实验

在网络上随机选取一段文本并随机插入不同数量的中文敏感词、无关字符干扰、中文拼音混合敏感词以及两种干扰方式混合的敏感词，然后使用本文提出的算法对该文本进行检测过滤。

本文实验使用的原始中文敏感词库为 Github 提供的开源中文敏感词库，先对词库进行去重，得到的无重复的中文敏感词库内共有 1140 个敏感词，其中暴恐类 164 个，色情类 305 个，政治宗教类 336 个，虚假广告类 251 个，脏话类 84 个。

使用本文提出的算法进行实验：1) 对中文敏感词库进行扩充得到中拼混合敏感词库；2) 构建敏感词树；3) 预处理待检测文本；4) 敏感词检测过滤。实验结果见图 5、表 1、表 2。

### 4.2. 实验结果分析

本文提出的敏感词过滤算法能够检测过滤长度较短、敏感词的数量长度较小的文本，也能够检测过滤长度较长、敏感词的数量长度较大的文本，即适用于文本长度以及敏感词长度数量不确定的情况。

由上述实验可知，本文提出的敏感词过滤算法对中文敏感词以及干扰“伪装”后的敏感词均有较好的过滤效果且并不会破坏文本中的对非敏感内容。

使用本文算法对含有不同数量的带有干扰的敏感词的文本的进行实验时，由表 2 的数据可知，文本

较短且包含敏感词数量较少时，算法的查全率和准确率可以达到 100%，当待检测文本字符数以及所包含的敏感词数增大时，准确率维持在 100%，查全率稍有降低，在最极端即文本内容全为敏感词的情况下，本文算法的查全率为 95.3%，故在其他情况下本文算法的查全率要高于 95.3%。

原文：在约180万现有吸毒人员中，滥用合成@毒#%品的吸!毒人员约103万名，滥用阿pian类毒品的吸毒人员约73万名，海luo因、冰&*毒、可卡因等滥用品种仍维持较大规模，dama吸食人数逐年上升，新精神活性物质滥用时有发现，花样不断翻新，包装形态不断变化，有的甚至伪装成食品饮料，出现“毒邮票”、“毒糖果”、“毒奶茶”，极具伪装性、隐蔽性、诱惑性。疫情防控下，常见du+=品难以获取，吸毒人员转而寻求其他物质替代，各地查处滥用du冷ding、安眠$#&tong等管制药物，吸食含合成大麻素、“xiao气”、氯胺酮等替代物质情况增多，一些大城市出现“xiniu液”、“零号jiaonang”等色胺类物质的毒品。
相匹配的词：['吸毒', '毒品', '吸毒', '阿pian类', '毒品', '吸毒', '海1uo因', '冰毒', '可卡因', 'dama', 'du品', '吸毒', 'du冷ding', '安眠tong', '大麻', 'xiao气', '氯胺酮', 'xiniu液', '零号jiaonang', '色胺类', '毒品'] 21
过滤结果：在约180万现有**人员中滥用合成**的**人员约103万名滥用********的**人员约73万**********等滥用品种仍维持较大规模****吸食人数逐年，上升新精神活性物质滥用时有发现花样不断翻新包装形态不断变化有的甚至伪装成食品饮料出现毒邮票毒糖果毒奶茶极具伪装性隐蔽性诱惑性疫情防控下常见***难以获取**人员转而寻求其他物质替代各地查处滥用**********等管制药物吸食含合成**素********等替代物质情况增多一些大城市出现**********等***物质的**

**Figure 5.** Results of sensitive word filtering experiment
**图 5.** 敏感词过滤实验结果

**Table 1.** Experimental results of vocabulary expansion and word tree construction
**表 1.** 词库扩充和词树构建实验结果

| | 原始中文敏感词库 | 中文拼音混合敏感词库 | 敏感词树 |
|---|---|---|---|
| 个数(分支数) | 1140 | 9800 | 679 |
| 运行时间 | | 28.82 s | 0.031 s |

**Table 2.** Results of sensitive word filtering experiment
**表 2.** 敏感词过滤实验结果

| 文本字符数 | 文本包含的敏感词个数 T | 算法过滤的敏感词个数 M | 算法正确过滤的敏感词个数 N | 运行时间 | 查全率 R | 准确率 P | 综合评价指标值 F |
|---|---|---|---|---|---|---|---|
| 261 | 21 | 21 | 21 | 0.016s | 100% | 100% | 1 |
| 6914 | 1000 | 971 | 971 | 0.035s | 97.1% | 100% | 0.985 |
| 25165 | 9800 | 9445 | 9339 | 0.759s | 95.3% | 100% | 0.982 |

## 4.3. 算法复杂度分析

1) 中文拼音敏感词库扩充算法的时间复杂度。

本文中文拼音敏感词库扩充算法原理为排列组合，其时间复杂度与原始中文敏感词库中敏感词个数和长度有关，即时间复杂度为$O\left(n\times 2^{\overline{len}}\right)$，其中 $n$ 为原始中文敏感词库中敏感词个数，$\overline{len}$ 为原始中文敏感词库中敏感词的平均长度。

2) 敏感词树构建算法的空间复杂度。

本文使用字典树结构构建敏感词树，构建时将整个拼音看作一个字符存储在一个节点中，如图 6，对比原始存储方式能够进一步节省存储空间，降低空间复杂度。

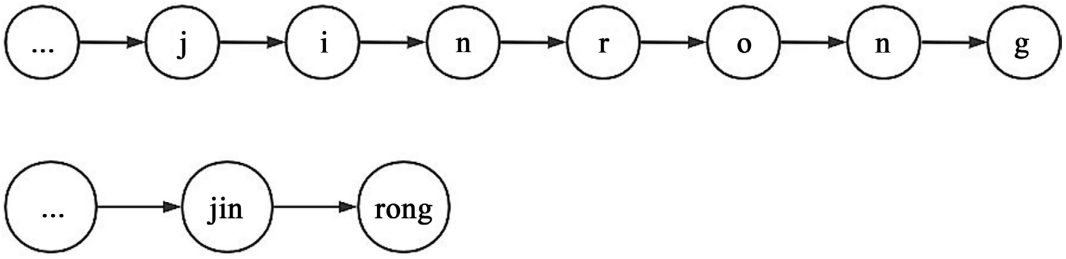


**Figure 6.** Comparison of branch structure of sensitive word tree
**图 6.** 敏感词树分支结构对比

本文敏感词树构建算法的空间复杂度为$O(N\times L)$ [10]，其中 $N$ 为敏感词树节点数量，$L$ 为节点长度。

3) 敏感词树构建算法的时间复杂度。

由于单个拼音存储在一个节点中，本文敏感词树构建算法的时间复杂度为$O\left(n'\times\overline{len'}\right)$，其中 $n'$ 为中文拼音混合敏感词库中敏感词个数，$\overline{len'}$ 为敏感词的平均长度。

4) 敏感词过滤算法的时间复杂度。

本文敏感词过滤算法的检测过滤时间与待检测文本的长度有关，即时间复杂度为$O(l)$，其中 $l$ 为待检测文本长度。

## 5. 结论

本文针对当前网络上通过各种干扰形式“伪装”的敏感词，提出了一种基于 DFA 的中拼混合敏感词过滤算法，扩充了原始中文敏感词库使得词库更加完善便于算法进行后续工作；对文本长度以及敏感词长度数量没有要求，能够有效地对检测文本中的中文敏感词以及干扰“伪装”后的敏感词进行检测过滤，且不会破坏文本中的非敏感内容，解决了一般的系统过滤方法难以成功检测过滤干扰“伪装”后的敏感词的问题；算法复杂度较低满足实际应用需求。

## 参考文献

[1] Liu, C., Wang, W.Y., Wang, M., *et al*. (2017) An Efficient Instance Selection Algorithm to Reconstruct Training Set for Support Vector Machine. *Knowledge-Based Systems*, **116**, 58-73. https://doi.org/10.1016/j.knosys.2016.10.031

[2] Guan, D.H., Yuan, W.W., Lee, Y.K., *et al*. (2008) Improving Supervised Learning Performance by Using Fuzzy Clustering Method to Select Training Data. *Journal of Intelligent & Fuzzy Systems*, **19**, 321-334.

[3] Xue, P.Q., Nurbol, and Wushour, I. (2016) Sensitive Information Filtering Algorithm Based on Text Information Network. *Computer Engineering & Design*, **37**, 2447-2452.

[4] Liu, B.G., Chen, Q.C. and Lei, X.F. (2020) Efficient Multi-Pattern Matching Algorithm for PDF Content Search. *Application Research of Computers*, **37**, 1755-1759.

[5] Chen, Y.J., Wushour, S. and Yu, Q. (2019) An Improved Multi-Pattern Matching Algorithm based on Aho-Corasick Algorithm. *Modern Electronics Technique*, **42**, 89-93.

[6] 李丹阳, 赵亚慧. 基于字典树语言模型的专业课查询文本校对方法[J]. 延边大学学报, 2020, 46(3): 260-264.

[7] 蒋琳, 徐颖. 基于 DFA 访问结构的多授权机构 ABE 方案设计[J]. 无线电工程, 2022, 52(8): 1302-1309.

[8] 周永福, 曾志. 基于 DFA 算法的政务云敏感词汇监测系统实现[J]. 科技与创新, 2022(20): 152-155.

[9] 孙芳媛. 基于倒排索引和字典树的站内搜索引擎的设计与实现[D]: [硕士学位论文]. 哈尔滨: 哈尔滨工业大学, 2016.

[10] Wang, M.H. and Hung, C.P. (2003) Extension Neural Network and Its Applications. *Neural Networks*, **16**, 779-784. https://doi.org/10.1016/S0893-6080(03)00104-7