Learning Regular Expressions for Interpretable Medical Text Classification Using a Pool-based Simulated Annealing and Word-vector Models

Chaofan Tu • Ruibin Bai • Zheng Lu • Uwe Aickelin • Peiming Ge • Jianshuang Zhao

Abstract We considered a rule-based engine consisting of auto-generated regular expressions for high quality and fully interpretable medical data processing. Existing Deep Neural Network (DNN) methods often present high quality performance in most Natural Language Processing (NLP) applications. However, these methods are often served as “black-box” tools and resulting solutions lack sufficient interpretability, which is a critical requirement for automated medical data processing. There have been some efforts in combining these DNN approaches with rule-based methods (e.g. regular expressions) to improve their accuracy and interpretability but construction of these regular expressions can be extremely labour intensive for large datasets. This research aims to reduce those manual efforts while maintaining high quality medical data processing and interpretability. A modified Simulated Annealing method, called pool-based simulated annealing with word vector model, or PSAW, is proposed to automatically derive regular expressions to form a rule-based classification engine. The
proposed method was tested for 30 medical text classification tasks, each containing half a million real-life data from one of China’s largest online medical platforms, the proposed PSAW method showed promising potentials compared to domain experts. Additional results show that the performance of our method can be further improved by combining it with the state-of-the-art DNN approaches. We believe the proposed method can be a vital complementary tool for text classification applications which require high levels of interpretability of the solutions.

1 Introduction

With the prevalence of modern computerised technologies, the chances to boost the accuracy of the auxiliary diagnosis models are manyfold which in turn enhances the doctor/hospital operational efficiency. By using latest technologies in speech recognition, machine vision, Natural Language Processing (NLP), machine learning, and others, data mining is becoming a necessity in the field of intelligent healthcare sector where a huge amount of digital data is available. For instance, the IBM Watson for ontology has demonstrated concordance rates of 96% for lung, 81% for colon, and 93% for rectal cancer cases with a multi-disciplinary tumour board in India [1]. The IBM Watson collects the data from 8500 hospitals, insurers, and government agencies [2]. Another popular application of intelligent healthcare is the DeepMind’s Streams medical diagnosis mobile application. The application sends alerts to nurses and doctors when a patient’s readings look abnormal through analysing the medical records of 1.6 million patients [3]. The availability of vast quantity of digitised healthcare and patient data plays an important role in the auxiliary diagnosis models.

This work is based on the collaboration with a very large online medical platform that allows patients to see a doctor online. The average number of daily consultation requests submitted to this platform in 2017 exceeded 370,000 [4]. In order to assign appropriate doctors across the different disciplines such as gynaecology, paediatrics, dermatology and so on, the system has to first deal with the classification of medical inquiries, which are sentence-level patients’ complaints in text format. Any mistakes in the classification process will lead to doctor miss-assignment and therefore reduce the system overall efficiency, especially in this real-time online scenarios. For the sake of not only reducing the number of employees who should be available 24/7 handling the medical reception but also enhancing the platform’s operational efficiency, it is very important to implement an automated classification system.

Some state-of-the-art approaches such as DNN have provided good performance in various NLP tasks while no human effort is required once a suitable model has been built, but their main drawback is the lack of interpretability for medical use in such “black box” approaches. Additionally, it may even require complete retraining when one particular classification task is updated. Therefore, the medical industry prefers to ensure good interpretability of system for rigorous validation requirements even if additional professional human resources are required.

It is impossible to deal with massive classification tasks by manual confirmation. Regular expressions are widely used in text-matching techniques that are fully interpretable compared to learning-based models. The regular expression based classification engine constructed by domain experts can be used to verify the most possible results from DNN model, meanwhile not only enhancing the interpretability and improving the precision of entire classification system. The rules engine plays an important role especially for classification tasks that are particularly challenging for DNN. Its main advantage is its shortcomings: manual construction of the rules require domain knowledge and a considerable amount of human work.

In order to achieve a good trade-off between interpretability and human efforts for the medical text classification problem, this paper proposes a fully automated system for learning regular expressions to capture relevant words and synonyms, discard irrelevant words and form good word sequences for good readability and further revision. The contributions of this paper are as follows:
• A specially designed structure of solutions is proposed to reduce the complexity whilst
  maintaining flexibility;
• A method called PSAW is proposed, combining pool-based Simulated Annealing and
  word-vector model to enhance the readability of auto-generated regular expressions.
• Impacts of parallel and iterative strategies for tasks of learning regular expressions
  have been intensively explored by comparing two extended versions of PSAW.
• On real industry datasets, the performances of PSAW and domain experts were
  compared.

2 Related works

Text classification involves assigning a text document to a set of pre-defined classes
automatically. Classification is usually done by using significant words or features extracted
from the raw textual document. This is a typical supervised machine learning task with
predefined classes and training text documents with class labels [5]. Automated text
classification usually includes steps such as pre-processing (eliminating stop-words, etc.),
feature selection using various statistical or semantic approaches, and text modelling [5]. Until
late 80’s, text classification task was based on Knowledge Engineering (KE), where a set of
rules were defined manually to encode the expert knowledge on how to classify the documents
given the categories [6]. Since there is a requirement of human intervention in knowledge
engineering, researchers in 90’s have proposed many machine learning techniques to
automatically manage and organise textual documents [6]. The accuracy of those machine
learning techniques is comparable to that of human experts and no artificial labour work from
either knowledge engineers or domain experts is needed for the construction of a document
management tool [7].

Text classification involves challenges and difficulties. First, it is difficult to capture high-
level semantics and abstract concepts of natural languages. Second, semantic analysis, a major
step in designing an information retrieval system, is not easy to be well understood by
machines. Third, the high dimensionality (e.g. thousands of feature vectors) of data poses
negative influences for classification tasks [8].

In order to perform text classification, we first need to use a proper text representation.
Bag of Words (BoWs) is one of the most commonly used methods to represent a document. By
using a fixed global vocabulary, the BoWs use a vector to represent a text document based on
the frequency count of each term in the document. This kind of methods of text representation is
called Vector Space Model (VSM) [9]. Unfortunately, BoWs/VSM representation scheme has
its own limitations such as very high dimensionality of the representation, loss of correlation
with adjacent words, and absence of semantic relationship [10]. Another VSM-based method is
a neural network model called Word2vec which is used in this paper for extracting word
embeddings, which was proposed by Mikolov et al. in 2013 [11, 12]. This kind of fixed-length
vector representation (often hundreds of dimensions) trained by deep learning model has shown
the ability to carry semantic meanings. This technique can be used in various NLP tasks such as
text classification, speech recognition, and image caption generation [13].

After text presentation, word embeddings or numerical representations for text feature
extraction can be fed into plain classifiers like the Naïve Bayes, decision tree, neural network,
support vector machine, hybrid approaches etc. [8]. The Naïve Bayes classifier is the simplest
probabilistic classifier used to classify the text documents into predefined labels [8]. The K-
Nearest Neighbour classifier is a non-parametric method which can be shown that for large
datasets the error rate of the 1-Nearest Neighbour classifier is not likely to be larger than twice
the optimal error rate [8]. Centroid based classifier is a popular supervised approach used to
classify texts into a set of predefined classes with relatively low computation [8]. Decision trees
are the most widely used inductive learning methods [8]. Decision trees’ robustness to noisy
data and their capability to learn disjunctive expressions seem suitable for document
classification [8]. Support Vector Machine (SVM) is a supervised classification algorithm that
has been extensively and successfully used for text classification tasks [8]. Neural network
based text classifiers are also prevalent in the literature, where the input units are denoted as feature terms, the output unit(s) is the category or categories of interest, and the weights on the edges connecting units form dependence relations [8].

With a simple convolutional neural network (CNN) built on top of word-vector models, a series of experiments of sentence-level text classification problems suggest that unsupervised pre-training of word vectors is an important ingredient in deep learning for NLP [14]. Neural network based approaches are strong in terms of precision and recall but usually less interpretable because those “black box” models cannot be logically explained [15]. In addition, those “black box” approaches cannot be quickly modified except retraining the whole neural network models [16]. To address those difficult issues, some related work has shown that regular expressions can be effectively used to solve text classification problems in an interpretable way.

A novel regular expression discovery (RED) algorithm and two text classifiers based on RED were designed to automate both the creation and utilisation of regular expressions in text classification [17]. The proposed RED+ALIGN method correctly classifies many instances that were misclassified by an SVM classifier. A novel transformation-based algorithm, called ReLIE, was developed to learn such complex character-level regular expressions for entity extraction tasks and related experiments demonstrated that it is effective for certain classes of entity extraction in text documents [18].

Automated regular expression generation can also be viewed as a data-driven optimisation problem. In this paper, a well-known simulated annealing hyper-heuristic [19] has been adapted for learning regular expression based classifiers for text classification. The choice of this approach is based on the fact that there are naturally multiple neighbourhood operators available for generating regular expression variants and hyper-heuristics can learn to orchestrate the selections of different operators to achieve high performance across different problems. It has been shown that specially designed neighbourhood operators of SA will lead to better performance [15].

3 Problem description

Formally the problem can be defined as follows: given a set of predefined classes $C$ (or medical templates in the context of our application) and a set of text inquiries $Q$, the problem is to classify each inquiry $q \in Q$ to one of the classes $c \in C$ automatically based on a set of previously labelled examples by medical experts. Table I shows examples of the classification, text inquiry is usually a piece of text information given by the user, describing the medical conditions or problems; the classification task is to select the most appropriate medical template for this inquiry.

| Text inquiry                                                                 | Medical template       |
|------------------------------------------------------------------------------|------------------------|
| “My girl is three years old and always coughs without fever, what can I do for her?” | Cough: 1-3 years old child |
| “I have been suffered from pain in my lower abdomen for 3 weeks.”          | Adult bellyache         |
| “The acne grows on the back, and recently it is a little itchy near it.”    | Folliculitis            |
| “I have a serious hair loss, how to make hair?”                            | Hair loss               |

Let $R$ be a regular expression designed for the classification of class (or medical template in our application) $C$ (denote $|C|$ be the number of inquiries in class $C$), and let $M(R, Q) \in Q$ be the set of all medical texts matched by $R$ in $Q$; Denote $M_p(R, Q) = \{q \in M(R, Q): q$ is an instance of $C\}$ to the set of all correctly matched entries (medical text inquiries) and denote $M_n(R, Q) = \{q \in M(R, Q): q$ is not an instance of $C\}$ is the set of all mismatched entries (medical
text inquires). Like many classification problems, this problem also has two performance indicators, which are precision and recall as bellows:

\[
\text{precision}(R, Q) = \frac{|M_p(R, Q)|}{|M_p(R, Q)| + |M_n(R, Q)|}
\]

(1)

\[
\text{recall}(R, Q) = \frac{|M_p(R, Q)|}{|C|}
\]

(2)

The well-known F-measure (also called F-score) can be a better single metric when compared to precision and recall. Given a non-negative real \(\beta\) for users' preference, it can be expressed as:

\[
F_\beta(R, Q) = \frac{(1+\beta^2) \times \text{precision}(R, Q) \times \text{recall}(R, Q)}{\beta^2 \times \text{precision}(R, Q) \times \text{recall}(R, Q)}
\]

(3)

The problem of automated learning of regular expression based classifiers for medical text in this paper can be formally expressed as an optimization problem for regular expression \(R\). Let \(S\) be the solution space of \(R\), for a given class of \(C\) and labelled dataset \(W\) which can be divided into a positive part and a negative part, the problem is to find a solution with the optimal objective function \(F_\beta\) from the solution space \(S\). So this problem can be defined as:

\[
R_{\text{target}} = \arg \max_{R \in S} F_\beta(R, Q)
\]

(4)

4 Methodology

4.1 Proposed Structure

In this problem, each solution is encoded as a vector of \(m\) regular expressions <\(R_1, R_2, ..., R_m\)> . To check whether a particular inquiry belongs to a class (or template), the regular expressions in the vector is executed one by one sequentially in the same order of the vector for the inquiry under consideration. If the inquiry is matched by any of regular expressions, the inquiry is said to be in the class, otherwise, it is not in the given class. Each regular expression \(R_i\) is derived via a combination of functions and terminals defined in Table I and follows a global structure of two parts \(P_i\) and \(N_i\) concatenated by the NOT function #_#, where \(P_i\) tries to match all positive inquiries and \(N_i\) is then used to filter out the list of falsely matched inquiries by \(P_i\). That is, each regular expression has the following format:

\[
R_i = (P_i).(#_#(N_i))
\]

(5)

| Name     | Label | Description                        |
|----------|-------|------------------------------------|
| NOT      | #_#  | Function to negate a given expression. |
| Sub-expression | e   | An expression or term obtained through a combination of words and functions listed below. |
| OR       | | Function to test logic or of two expressions. |
| Words    | w    | List of keywords extracted from the target text set. |
| AND      | | Function to test logic AND of two expressions. |
| Adjacency | \{a, b\} | Function to test whether the distance between two words \(w_1\) and \(w_2\) are in the range \([a, b]\), which is an extended function of the AND function. |

TABLE II. FUNCTIONS AND TERMINALS

For the purpose of achieving better readability of regular expressions \(R_i\), and limiting the size of search space, the following constraints are applied to each of regular expressions \(R_i\):

1) Each regular expression \(R_i\) has at most one NOT function.
2) The positive part \(P_i\) or negative part \(N_i\) is only composed of OR functions and it is defined as the outer OR structure as below:
\[ R_i = (e_{p1}|e_{p2}|...|e_{pm}), (\#\#(e_{a2}|e_{a3}|...|e_{am})) \]  

3) Function \textit{OR} can be used in any sub-expression \( e \) and it is defined as the \textit{inner OR structure}, but there should not be any other nested functions except \textit{OR} in \( e \). That is, the sub-expression \( w_j|(w_2|w_j) \) is acceptable because the nested function is \textit{OR} but the sub-expression \( w_j|(w_2|w_j) \) is not permitted since the nested function is \textit{AND}.

4) Function \textit{AND} used in any sub-expression \( e \) can contain nested functions of both \textit{AND} and \textit{OR}. For example, both the sub-expression \( w_j|(w_2\cdot w_3) \) and the sub-expression \( w_j|(w_2|w_j) \) are acceptable.

The \textit{outer OR structure} is used to directly compose positive part \( P_i \) or negative part \( N_i \) according to condition 2, while the \textit{inner OR structure} used in any sub-expression \( e \) cannot contain any nested function except \textit{OR} due to condition 3. Through the above restrictions, the overall structures of regular expressions \( R_i \) have been limited to a maximum of two levels of nested \textit{OR} structures.

Corollary 1 With the same terminals and functions listed in Table II, there always exists one or more regular expressions that satisfy all the above conditions and is equivalent to any expression without any constraint of structures.

Proof: For condition 1, it’s obviously because the \textit{NOT} function is essentially one kind of set operation, multiple \textit{NOT} functions can be reduced to one finally;

For condition 2, it’s evident that any other single or multiple functions can apply \textit{OR} function to the outer layer of itself, because the \textit{OR} function has a lower priority than any other function except \textit{NOT};

For condition 3 and 4, if a sub-expression \( e \) of \( P_i \) is directly composed of \textit{OR} function and the \textit{OR} function contains an \textit{AND} function, \textit{Pi} can be clearly transformed into a new expression which meets the condition 2 as below:

Let \( e_{p2} = w_j|(w_2\cdot w_3) \),

\[ P_i = (e_{p1}|(w_j|w_2\cdot w_3)|...|e_{pm}) = (e_{p2}|w_j|w_2\cdot w_3|...|e_{pm}) \]  

(7)

If a sub-expression \( e \) of \( P_i \) is composed of one \textit{AND} function and the \textit{inner OR structure} contains an \textit{AND} function such as \( (w_j|(w_2\cdot w_3)) \cdot w_4 \), \( P_i \) can be also transformed into a new expression which meets the condition 2 and 4 as below:

Let \( e_{p2} = (w_j|(w_2\cdot w_3))\cdot w_4 \),

\[ P_i = (e_{p1}|((w_j|(w_2\cdot w_3))\cdot w_4)|...|e_{pm}) = (e_{p2}|w_j\cdot w_3(w_2\cdot w_3)\cdot w_4|...|e_{pm}) \]  

(8)

End of proof. That is, although we restrict the possible formats of our regular expressions to one kind of two-layer nested structures, their expressiveness is not reduced. These conditions not only simplify the structure of solutions but also contribute to enhancing the readability and interpretability.

4.2 Solution Pool Mechanism

According to problem descriptions and structures defined above, the medical text classification problem in this paper is transformed into a combinatorial optimisation problem. The Simulated Annealing algorithm is a large-scale combined problem global optimisation algorithm, which is widely used to solve the NP-hard combination optimisation problem.

An improved Simulated Annealing method is applied as the evolutionary computation algorithm to solve the problem in this paper, with a well-designed solution pool mechanism to enhance the diversity of solutions as shown in figure 1:
The number of solutions in the elite solution pool is set to a fixed value, and the same amount of new solutions are transformed from an initial solution for the initialization of the elite solution pool.

The number of solutions in the neighbour solution pool is the same as the elite solution pool. Each solution in the elite solution pool will produce a new solution in each iteration during the entire period, then a totally updated neighbour solution pool can be formed from those all newly-generated solutions.

The elite solution pool will be also updated in each iteration during the entire evolution. For each solution in the updated neighbour solution pool, one solution in the elite solution pool will be randomly selected for comparison and replacement. The used acceptance criterion for replacement adopts Metropolis criterion based on Simulated Annealing algorithm.

Every time the best solution in the elite solution pool is always retained during the whole period. The details of the proposed solution pool mechanism are shown below in figure 2.

**Set** the capacity of a solution pool to be $N_{pool}$;
**Define** a set of each elite solution $S_{ei}$ ($i = 1, ..., N_{pool}$) as the elite solution pool $P_e$;
**Define** a set of each neighbour solution $S_{nj}$ ($j = 1, ..., N_{pool}$) as the neighbour solution pool $P_n$;
**Set** the best solution in $P_e$ as $S_{e\_best}$;

Solution Replacement:

begin
    $j = 1$
    while ($j < N_{pool}$) do
        select $S_{nj}$;
        select $S_{ei}$ randomly from $P_e$;
        let $\delta$ be the difference in the evaluation function between $S_{ei}$ and $S_{nj}$;
        if the Metropolis criterion of simulated annealing is satisfied by $\delta$
            $S_{ei} = S_{nj}$
        endif
        $j ++$
        add $S_{e\_best}$ into $P_e$ temporarily;
        sort $P_e$ by the evaluation function of each solution $S_{ei}$ ($i = 1, ..., N_{pool} + 1$);
        $P_e = \{ S_{e_{i}} \mid i = 1, ..., N_{pool} \}$
        $S_{e\_best} = S_{e_{i}}$
    end

Fig. 2. Pseudo-code of the proposed solution pool mechanism

4.3 Initialisation

From the above description, an existing initial solution is a precondition for the initialisation. In order to balance speed and readability, we carry out a method using word frequency and similarity comparison to quickly generate a group of keywords as an initial solution. The specific steps are described below:

1) First, if the frequency of a word in the positive dataset exceeds the predetermined times defined as $TD_F$, this word will be added into the set of keywords;
2) Second is sorting the keyword set by word frequencies and calculating cosine similarities between any two different words’ vectors in the top $N_W$ number of keywords. If the cosine similarity exceeds the predetermined value defined as $TD_S$, these two words are considered as a group of same subject words;

3) Third, one group of subject words is randomly selected and those words are connected as an inner OR structure as an initial solution with an empty negative part. Below is an example:

\[
\text{(headache | dizzy | giddy | dizziness)}\).
\]

One or more regular expressions generated by the above steps from the initial solution $S_{\text{init}}$.

4.4 Neighbourhood Operators

In our proposed method, it is decided by a random strategy firstly whether to update the positive or negative part of solutions, then seven specially designed neighbourhood operators are used for generation of a new solution.

O1: **Adding OR type 1** is an operator to add a word to the inner OR structure. First randomly select ten words from the set of positive words or the set of negative words. Then randomly select an existed word from the inner nested OR structure, and calculate the cosine similarity $Sim_i (i = 1, \ldots, 10)$ between the existed word and the other ten words based on pre-training word vectors. Finally, only one word of the ten will be selected to add to the inner OR structure with the largest probability $Prob_i$ as below:

\[
Prob_i = \frac{Sim_i}{\sum_{i=1}^{10} Sim_i}
\]  

The extension of the inner OR structure by this type of function combines the information of the pre-trained deep learning model, word2vec, so that the readability of regular expression has been considered during the evolutionary.

O2: **Adding OR type 2** is an operator to add a sub-expression to the outer OR structure. First, randomly select a word from the positive or negative word set. If the selected word does not exist in the outer OR structure, add this word into it; if the word is already in the outer OR structure, then randomly select another word to form a non-repeating AND (or Adjacency) sub-expression $e$ and finally add $e$ into the outer OR structure.

O3: **Removing OR** is an operator to randomly delete a sub-expression that makes up the outer or inner OR structure as inverse operations of O1 and O2.

O4: **Adding AND** is an operator to extend the AND (or Adjacency) structure in the sub-expression in the outer OR structure. Randomly pick a word to insert into an existing AND (or Adjacency) structure or form a new AND structure with an existing word.

O5: **Swap** is an operator to exchange the positions of any two sub-expressions or words in the AND (or Adjacency) structure.

O6: **Distance** is an operator to randomly change the maximally permitted distances between two expressions or words based on a given Distance Table. So the AND function can be considered as an Adjacency structure with unrestricted distance.

O7: **Removing AND** is an operator to randomly delete one sub-expression $e$ or word $w$ that makes up one AND (or Adjacency) structure, as an inverse operator of O4.

4.5 Solution Decoding and Evaluation

Each regular expression $R_i$ in solution pools should be decoded to a valid regular expression that can be passed through the general regular expression matching engine. There are two main points to note here. The logical symbols defined in this paper are not exactly the same as the symbolic system of regular expressions; The **NOT** function defined in our system does not exist in regular expression matching engine, so the positive and negative parts of $R_i$ need to be separately handled. An example of this conversion should be done as follows:
The positive part of the converted $R_i$:

\[.*((w_1|w_2|w_3).*w_4|w_5.{0,10}(w_6|w_7|w_8).*(w_9|w_10|\ldots|w_p)).*.\]

The negative part of the converted $R_i$:

\[.*((w_1|w_2|w_3|w_4|w_5).{0,10}w_7|w_8|\ldots|w_n)).*.\]

The performance of each regular expression based solution or classifier will be evaluated based on the F-measure value according to the above description in section 2. In this paper, the parameter $\beta$ for F-measure is set to 0.2 for the purpose of giving more attention to precision.

4.6 Overall Algorithm

In this paper, a modified Simulated Annealing method, called pool-based simulated annealing with word vector model, or PSAW, is proposed to automatically derive regular expressions to form a rule-based classification engine. Below figure is the overall flow of PSAW.

![Flowchart of PSAW](image)

Before initialisation the pre-processing includes dividing the training data set into positive and negative sets, performing Chinese word segmentation, removing stop words, and pre-training the Word2vec model.

The initial solution has been generated after the initialisation. In the beginning, the elite solution pool will be fully filled with new solutions transformed by the initial solution.

According to the Metropolis criterion, solutions in the elite solution pool may be replaced by solutions in the neighbour solution pool. Parameters such as the temperature of classic simulated annealing would be updated in each iteration. The program terminates when the total iterations are finished or the stop condition is met.

4.7 Iterative and Parallel Strategies

To further explore the impacts of different running strategies in computational time and performance, we designed and implemented an extended version called PSAW-I for iterations strategy and another extended version called PSAW-P for parallelism.
1) PSAW-I: It is considered to improve the recall by learning regular expressions iteratively, that is, before learning the next classifier, those text inquiries matched by existing classifiers should be filtered first in the training set.

2) PSAW-P: Consider the method of pre-dividing the positive training dataset for parallel acceleration. When the last parallel task is terminated, all the sub-solutions are merged as one solution for the whole. Pre-dividing is based on semantic clustering on our datasets and also set random division as a comparison baseline.

5 Experiment

5.1 Data and Parameter Settings

Because of the collaboration with a very large online healthcare platform, the experiments in this paper are based on high-quality training and test data from the real production environment. The numbers of real-life medical text inquiries in training set and test set are 1,800,000 and 500,000 respectively. If not stated separately, the following parameters are used in this paper.

1) For the initialization: $TD_F = 5; N_w = 100; TD_S = 0.75$;
2) For the neighbourhood operators: $Distance Table = \{0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 100\}$;
3) For pool-based simulated annealing: starting temperature $T_S = 0.5$; stopping temperature $T_E = 0.05$; solution pool capacity $N_{pool} = 10$; Total iteration $K = 1000$;
4) For F-measure: $\beta = 0.2$.

5.2 Evaluation of Solution Pool Mechanism

In this experiment, we controlled the variables $N_{pool}$ and $K$ for learning one regular expression for the same template $C_I$ to evaluate the solution pool mechanism. The $N_{pool}$ of groups 1~3 were set to 1, 10, 50 and the $N_{pool}$ of group 4 was set to 1 to represent the classic simulated annealing without the proposed mechanism. The total number of new solutions in group 4 was set as the same as group 2.

In TABLE III the results of groups 1~3 show that the higher the $N_{pool}$, the more time cost and the better the performance of F-measure ($F_m$).

| Group | $N_{pool}$ | $K$    | $F_m$ | Time (min) |
|-------|------------|--------|-------|------------|
| 1     | 1          | 1000   | 0.60  | 8          |
| 2     | 10         | 1000   | 0.76  | 59         |
| 3     | 50         | 1000   | 0.77  | 364        |
| 4     | 1          | 10000  | 0.72  | 130        |

The comparison between group 4 and 2 shows that the solution pool mechanism significantly not only enhances performance but also reduces time cost. This is because the solution update process of group 4 would make its solution more and more complex to increase the evaluation time and this mechanism in group 2 can improve the diversity of solutions.

5.3 Comparison of Iterative and Parallel Strategies

We have tested the proposed PSAW and its two extended versions of PSAW-I, PSAW-P on six different medical text classes $C_I \sim C_P$. For further explorations, PSAW-P version adapted two different methods, that are separately clustering division and random division. All solutions were set to contain three regular expression based classifiers. One PSAW-P group was set to use the widely-used k-means clustering method to divide the training dataset into three different
subsets for parallel processing, while another PSAW-P group was set to use the random trisection as a comparison baseline.

In TABLE IV and V, the PSAW-P version with k-means clustering showed the highest level of precision and the least time cost; the PSAW-I version showed the highest level of recall. The original PSAW version presented the most time cost, while its average of F-measure value is the best.

| Classes | PSAW | PSAW-I |
|---------|------|--------|
|         | Precision | Recall | \( F_m \) | Time(min) | Precision | Recall | \( F_m \) | Time(min) |
| C_1     | 0.89    | 0.65   | 0.87     | 266       | 0.76    | 0.76   | 0.76     | 220       |
| C_2     | 0.69    | 0.41   | 0.68     | 335       | 0.56    | 0.50   | 0.50     | 300       |
| C_3     | 0.71    | 0.11   | 0.58     | 417       | 0.52    | 0.25   | 0.50     | 372       |
| C_4     | 0.93    | 0.33   | 0.87     | 394       | 0.81    | 0.78   | 0.81     | 333       |
| C_5     | 0.84    | 0.69   | 0.83     | 405       | 0.87    | 0.83   | 0.87     | 345       |
| C_6     | 0.93    | 0.61   | 0.91     | 396       | 0.84    | 0.65   | 0.83     | 239       |
| AVG     | 0.83    | 0.46   | 0.81     | 369       | 0.73    | 0.63   | 0.72     | 302       |

| Classes | PSAW-P (Clustering) | PSAW-P (Random) |
|---------|---------------------|-----------------|
|         | Precision | Recall | \( F_m \) | Time(min) | Precision | Recall | \( F_m \) | Time(min) |
| C_1     | 0.87      | 0.42   | 0.83     | 86        | 0.86     | 0.52   | 0.84     | 110       |
| C_2     | 0.72      | 0.29   | 0.68     | 119       | 0.62     | 0.26   | 0.59     | 125       |
| C_3     | 0.71      | 0.11   | 0.59     | 138       | 0.66     | 0.09   | 0.54     | 144       |
| C_4     | 0.92      | 0.37   | 0.87     | 133       | 0.92     | 0.31   | 0.86     | 128       |
| C_5     | 0.89      | 0.54   | 0.87     | 136       | 0.86     | 0.51   | 0.84     | 134       |
| C_6     | 0.93      | 0.48   | 0.89     | 108       | 0.92     | 0.48   | 0.89     | 124       |
| AVG     | 0.84      | 0.37   | 0.80     | 120       | 0.81     | 0.36   | 0.77     | 128       |

### 5.4 Performance Distribution

The PSAW method has been further applied to learn regular expressions for 30 independent medical templates for evaluating its performances compared to medical experts. The following figure 4 is the precision and recall distributions of auto-generated classifiers by PSAW on the test dataset, compared to those manual regular expression based classifiers written by domain experts.

![Precision Distribution](image1)

![Recall Distribution](image2)

**Fig. 4.** Precision and Recall Distribution

The precision of most classifiers by domain experts exceeds 0.9 and auto-generated classifiers mostly exceed 0.8. The recall of most classifiers by domain experts exceeds 0.6, while the distribution of auto-generated classifiers is more uniform. The reason may be the parameter for the evaluation function was set to pay more attention to precisions rather than recalls (\( \beta = 0.2 \)).
5.5 Practicality Evaluation

Those auto-generated regular expressions can be used as secondary verification for fully interpretable medical data processing. 50 manual classifiers made by medical experts and 50 auto-generated classifiers by PSAW are randomly selected for third-party practicality blind evaluation using the below score table in TABLE VI.

| Score | Descriptions                          |
|-------|---------------------------------------|
| 1     | Cannot be used                        |
| 2     | Can be used after a lot of revisions  |
| 3     | Can be used after some revisions      |
| 4     | Can be used after minor revisions     |
| 5     | Can be used directly without revisions|

The score distribution of third-party blind evaluation is shown below in Figure 5.

According to the third-party blind evaluation, most auto-generated classifiers by PSAW are well readable and can be applied to practical use after some or minor revisions, which benefits from both the proposed solution structures and the use of word vector model.

5.6 Improvement Evaluation

one of China’s largest online medical platforms, the proposed PSAW method showed promising potentials compared to domain experts. Additional results show that the performance of our method can be further improved by combining it with the state-of-the-art DNN approaches. We believe the proposed method can be a vital complementary tool for text classification applications which require high levels of interpretability of the solutions.

Through the auto-generated regular expression based classifiers, a fully interpretable rule-based engine was derived for medical text classification. On the real medical industry dataset with different 140 text classification tasks, the rule-based engine was used to perform secondary verification of the most 5 possible results from a state-of-art DNN model that is being used in the very largest online medical platform of our collaborator. Below is the comparison of the new method of combining the rule-based engine with the DNN and the DNN in Figure 6.
The test showed that the combined use of the rule-based engine brought an average 9.35% improvement (from 74.94% to 84.29%) in precision compared to the only use of DNN model meanwhile ensuring good interpretability and full control of results. It can be clearly seen that the new method significantly reduced the number of tasks with low precision of less than 60% and significantly increased the number of tasks with high precision of more than 80%.

6 Conclusion and future research

In this work, construction of regular expressions for medical text classification is transformed into a combinatorial optimisation problem and solved a variant Simulated Annealing method, called, PSAW, which combines the classical simulated annealing with word vector model (pre-trained word2vec model). In addition, iterative and parallel strategies have been explored for further improvement in computational time and performance. Computational results on real-life instances show that PSAW is able to generate promising solutions. Although those auto-generated classifiers by PSAW cannot outperform experts’ classifiers on all instances, most of them are fully interpretable and can be served as drafts for domain experts for further improvements. We also integrated the regular expression classifiers with a state-of-art DNN classification model and tested them on real-life data. Considerable improvements were obtained.

Future research directions include using GPU to further speed up the algorithm, adoption of a multi-objective optimization model for regular expression generation, and theoretical analysis for more efficient regular expression encoding in the context of medical text classifications.

Acknowledgements This work is partly supported by the National Natural Science Foundation of China (Grant No. 71471092), Natural Science Foundation of Zhejiang Province (Grant No. LR17G010001), Ningbo Municipal Bureau of Science and Technology (Grant No. 2014A35006, 2017D10034).

References

1. IBM. Bringing confident decision-making to oncology. [cited 2018 24th June]; Available from https://www.ibm.com/watson/health/oncology-and-genomics/oncology/.
2. FOX, B.I. IBM Watson: Artificial Intelligence Meets Healthcare 2017 [cited 2018 24th June]; Available from https://www.asapnet.org/files/June2017/Presentations/ASAPJune17_Presentations07_Fox.pdf.
3. Ram, A. DeepMind develops AI to diagnose eye diseases. 2018 [cited 2018 24th June]; Available from: https://www.ft.com/content/84f6c6c-0787-11e8-9650-9e0ad2d7c5b5.
4. Limited, P.A.H.a.T.C. Introduction of Company. 2018 [cited 2018 24th June]; Available from: http://www.pahtg.com/media/1168/pagd-presentation-_cn-website-version-final-1.pdf.
5. Dalal, M.K. and M.A. Zaveri, Automatic text classification: a technical review. International Journal of Computer Applications, 2011. 28(2): p. 37-40.
6. Mitra, V., C.-J. Wang, and S. Banerjee, Text classification: A least square support vector machine approach. Applied Soft Computing, 2007. 7(3): p. 908-914.
7. Fung, G. P. C., Yu, J. X., Lu, H., & Yu, P. S., Text classification without negative examples revisit. IEEE transactions on Knowledge and Data Engineering, 2006. 18(1): p. 6-20.
8. Harish, B.S., D.S. Guru, and S. Manjunath, Representation and classification of text documents: A brief review. IJCA, Special Issue on RTIPPR (2), 2010: p. 110-119.
9. Salton, G., A. Wong, and C.-S. Yang, A vector space model for automatic indexing. Communications of the ACM, 1975. 18(11): p. 613-620.
10. Bernotas, M., Karklius, K., Lauritis, R., & Slotkienė, A., The peculiarities of the text document representation, using ontology and tagging-based clustering technique. Information Technology and Control, 2015. 36(2).
11. Mikolov, T., Chen, K., Corrado, G., & Dean, J., Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013.
12. 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 2013:p. 3111-3119.
13. Rong, X., word2vec parameter learning explained. arXiv preprint arXiv:1411.2738, 2014.
14. Kim, Y., Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882, 2014.
15. Moreo, A., Eisman, E. M., Castro, J. L., & Zurita, J. M., Learning regular expressions to template-based FAQ retrieval systems. Knowledge-Based Systems, 2013. 53: p. 108-128.
16. Dalianis, H., J. Sjöbergh, and E. Sneiders. Comparing manual text patterns and machine learning for classification of e-mails for automatic answering by a government agency. in International Conference on Intelligent Text Processing and Computational Linguistics. 2011. Springer. 2011:p.234-243.
17. Bui, D.D.A. and Q. Zeng-Treitler, Learning regular expressions for clinical text classification. Journal of the American Medical Informatics Association, 2014. 21(5): p. 850-857.
18. Li, Y., Krishnamurthy, R., Raghavan, S., Vaithyanathan, S., & Jagadish, H. V., Regular expression learning for information extraction[C]//Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2008: p. 21-30.
19. Bai, R., Blazewicz, J., Burke, E. K., Kendall, G., & McCollum, B., A simulated annealing hyper-heuristic methodology for flexible decision support. 4OR, 2012. 10(1): p. 43-66.
Author/s: Aickelin, U; Tu, C; Bai, R; Lu, Z; Ge, P; Zhao, J

Title: Learning Regular Expressions for Interpretable Medical Text Classification Using a Pool-based Simulated Annealing and Word-vector Models

Date: 2019

Citation: Aickelin, U., Tu, C., Bai, R., Lu, Z., Ge, P. & Zhao, J. (2019). Learning Regular Expressions for Interpretable Medical Text Classification Using a Pool-based Simulated Annealing and Word-vector Models. Ningbo, China

Persistent Link: http://hdl.handle.net/11343/239259

File Description: Accepted version