Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID-19. The COVID-19 resource centre is hosted on Elsevier Connect, the company's public news and information website.

Future Generation Computer Systems 98 (2019) 238–251

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Language model-based automatic prefix abbreviation expansion method for biomedical big data analysis

Xiaokun Du [a], Rongbo Zhu [a,*], Yanhong Li [a], Ashiq Anjum [b]

[a] College of Computer Science, South-Central University for Nationalities, Wuhan, China
[b] College of Engineering and Technology, University of Derby, Derby, UK

HIGHLIGHTS

- An automatic method is proposed for biomedical big data.
- This method overcomes the weaknesses of the traditional dictionary-based method.
- An optimization strategy is proposed for time complexity improvement.
- An excessive number of expansions causes poor abbreviation sense disambiguation.

ARTICLE INFO

Article history:
Received 29 October 2018
Received in revised form 15 December 2018
Accepted 13 January 2019
Available online 28 March 2019

Keywords:
Abbreviation expansion
Biomedical text analysis
Language model

ABSTRACT

In biomedical domain, abbreviations are appearing more and more frequently in various data sets, which has caused significant obstacles to biomedical big data analysis. The dictionary-based approach has been adopted to process abbreviations, but it cannot handle ad hoc abbreviations, and it is impossible to cover all abbreviations. To overcome these drawbacks, this paper proposes an automatic abbreviation expansion method called LMAAE (Language Model-based Automatic Abbreviation Expansion). In this method, the abbreviation is firstly divided into blocks; then, expansion candidates are generated by restoring each block; and finally, the expansion candidates are filtered and clustered to acquire the final expansion result according to the language model and clustering method. Through restrict the abbreviation to prefix abbreviation, the search space of expansion is reduced sharply. And then, the search space is continuous reduced by restrained the effective and the length of the partition. In order to validate the effective of the method, two types of experiments are designed. For standard abbreviations, the expansion results include most of the expansion in dictionary. Therefore, it has a high precision. For ad hoc abbreviations, the precisions of schema matching, knowledge fusion are increased by using this method to handle the abbreviations. Although the recall for standard abbreviation needs to be improved, but this does not affect the good complement effect for the dictionary method.

1. Introduction

Big data analysis has opened the door to a new era in biomedical fields, such as healthcare [1] and disease diagnosis [2,3], etc. Abbreviations are appearing more and more frequently in these areas, which significantly hinders development in related research fields such as biomedical text analysis [4,5], large biomedical ontologies [6]. Abbreviations are used in almost all types of data (structured, semi-structured, unstructured). Regarding unstructured data, Ammar et al. [7] have informed that English Wikipedia articles contain an average of 9.7 abbreviations per article and that more than 63% of the articles contain at least one abbreviation. At the sentence level, over 27% of sentences in news articles contain abbreviations. In the biomedical domain, the situation is becoming worse. Clinical narratives are typically produced under time pressure, which incites the use of abbreviations and acronyms [8]. A study focused on the electronic discharge summaries from a large, tertiary teaching hospital in Australia revealed that abbreviations were common, occurring at a frequency of one in five words [9]. For structural (e.g., relational databases, etc.) and semi-structural data (e.g., XML (Extensive Markup Language), knowledge graph etc.), abbreviations are widely used for element names since the identifier length is limited (for example, the identifier length is limited to less than 30 characters in Oracle). Abbreviations are substantial obstacles

* Corresponding author.
E-mail addresses: xiaokundu@mail.scuec.edu.cn (X. Du), rbzhu@mail.scuec.edu.cn (R. Zhu), liyanhong@mail.scuec.edu.cn (Y. Li), a.anjum@derby.ac.uk (A. Anjum).

https://doi.org/10.1016/j.future.2019.01.016
0167-739X/

in related fields such as ontology matching [10] and knowledge map construction.

Abbreviations can be separated into two classes: standard abbreviations and ad hoc abbreviations. The former are widely used and accepted, including acronyms such as HIV (Human Immunodeficiency Virus) and SARS (Severe Acute Respiratory Syndromes). The latter are abbreviations that are used in special situations. For example, in intensive care unit notes, sentences such as "61 y(year).o(old). M(Male) pt(patient) with a hx(history) of COPD (chronic obstructive pulmonary disease), HTN (Hypertension)…" often occur. Physicians use many ad hoc abbreviations under heavy time pressure. In order to understand above text, the corresponding expansions are chosen to replace the abbreviations (called text normalization). In the existing works [8,11,12], the operation of text normalization is divided into the following two steps: firstly, all possible expansions are selected from the dictionary; then, a suitable sense is selected according to the context to replace the abbreviations. If the abbreviation or the suitable expansion is not contained in the dictionary, above method could not work. This seriously hinders the effect of text normalization especially for the data with many ad hoc abbreviation such as EMR (Electronic Medical Record), structural and semi-structural data. If the expansion of the abbreviation could be generated by an automatic method, the effect of text normalization could be improved dramatically.

Abbreviation is a short form of a word or phrase. It is comprised of certain characters in the original order in word or expression. In the existing works, the LCS (Longest Common String) rule is used as the basic criterion to judge the correct expansion. A direct strategy to generate the expansion is to list all the candidate words or phrases meet the requirements of LCS. This strategy has the following drawbacks

1. The search space is difficult to be determined. The search space consists of all words and phrases. But getting all the phrases is impossible.

2. LCS is a loose rule for expansion, so not all the words or phrase meet LCS are rational expansion. For example, "MODS" is the abbreviation of phrase "Multiple Organ Dysfunction Syndrome". But it could be the abbreviation of "modest", "normal goods" etc. even if they meet LCS rule.

3. The expansion result will contain many phrases with similar semantics. For example, "deoxyribose nucleic acid" and "deoxyribose nucleic acids" will be generated for "DNA". This will result in the number of expansion in the result is very large. Subsequent operations would be impacted.

In this paper, a new method called LMAAE (Language Model-based Automatic Abbreviation Expansion) is proposed to get the expansion automatically. In LMAAE, the above three drawbacks are processed separately. The flow diagram of LMAAE is depicted in Fig. 1.

The procedure of LMAAE is divided into three steps: partition, expansion and filter, cluster. In each step, appropriate measures are proposed to alleviate the corresponding drawback. Details as follows:

Partition: In this step, the abbreviation is partitioned into several blocks. Each block is corresponding to a word of the phrase. For an abbreviation with n characters, the number of different partition is 2^n. In this paper, there are two means to reduce the number of partition. At first, the maximum number of words in phrase is limited according to the statistic of the phrase in abbreviation dictionary. And then, the rationality of partition is proposed to filter the partition. Through analysis of the number of rational block, the number of rational partition is restricted to a constant value.

Expansion and filter: For each block, all the words in the dictionary meets LCS rule consist of the expansion set of the block. And then, the expansion of the abbreviation is made up of the Cartesian product of each block. At first, through analysis of the characteristic of the abbreviation in dictionary, more than 99% of the abbreviation meet prefix abbreviation rule: abbreviation consists of the prefixes of the key words of the phrase. So prefix abbreviation rule is adopted to reduce the search space of expansion in this paper. This greatly improves the accuracy of the expansion. Secondly, not all the word sequence meet prefix abbreviation rule is a meaningful phrase, so the language model is used to evaluate and filter each word sequence to further reduce the expansion result.

Cluster: The output of previous step is a set with many expansions. But there are many phrases in the set with the same meaning. The mean-shift clustering algorithm is selected to cluster the results to eliminate redundant expansions.

Through above optimizing process, most of the disturbance terms are filtered from the result. The experiment result shows that the expansion set could include most of the expansion in the dictionary. The contributions of this paper can be summarized as follows.

1. For the first time, an automatic method is first proposed to generate expansion for the abbreviation not contained in dictionary. Collaborated with dictionary-based method, the LMAAE could improve the effective of text normalization dramatically.

2. Theoretical analysis of the time complexity of the automatic expansion method is presented, and several optimizing processes are proposed to improve the effective of expansion set and the time complexity of method.

3. An excessive number of expansions will cause poor ASD (abbreviation sense disambiguation) performance. Therefore, we introduce a new strategy for clustering expansions with similar semantics to improve the results.

The remainder of this paper is organized as follows. Section 2 introduces the related research work. The definitions related to abbreviations and expansions are provided in Section 3, as well as some preliminaries. In Section 4, an analysis of the time complexity of the LMAAE method is presented and a corresponding optimization method is suggested to improve it. To validate the LMAAE method, several experiments were conducted, as discussed in Section 5. Section 6 provides the conclusions and describes areas requiring further work.

2. Related work

In biomedical big data analysis, the main research achievements involve dictionary construction and ASD. This section provides brief summaries of these areas and introduces some other important fields related to abbreviations.

2.1. Abbreviation dictionary construction

At present, there are many public abbreviation dictionaries, such as AllAcronym,[1] Abbreviations,[2] and Stedman [13]. Through continuous improvement for almost 20 years, the number of dictionary entries has reached over one million, and the entries are checked manually one by one. Although these dictionaries are widely used in natural language processing, the shortcomings are obvious. Firstly, the maintenance cost is very high, and secondly, such dictionaries are ineffective for ad hoc abbreviations and some specific abbreviations such as eDNA (which consists of a word "extracellular" and an abbreviation "DNA" (Deoxyribo Nucleic Acid)).

[1] https://www.allacronyms.com/.

[2] https://www.abbreviations.com/.

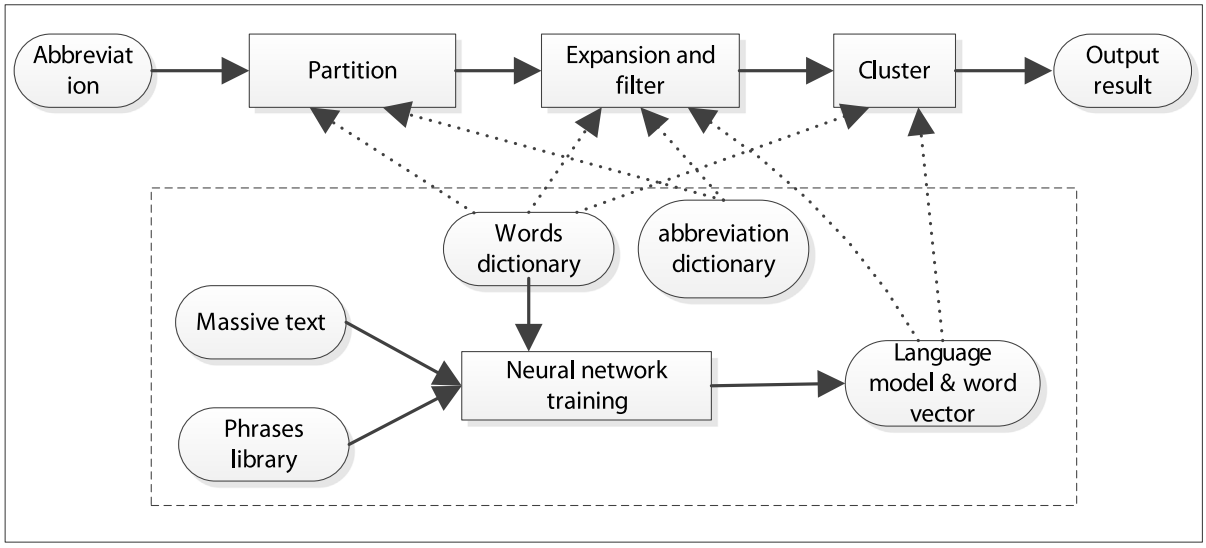


Fig. 1. Flow diagram of LMAAE.

An abbreviation dictionary contains pairs of abbreviations and their expansions in the form <abbreviation, expansion>, for example, <SARS, severe acute respiratory syndrome>. In recent years, numerous methods have been suggested to find expansions automatically. Most methods can be classified into two categories: pattern-matching techniques and machine learning-based methods. For pattern-matching techniques, the LCS is the most important standard for candidate pair judgment. Representative works include those on an acronym-finding program [14] and three-letter acronyms [15]. Because the LCS restriction is so loose that many intrusive expansions are introduced, the search range and abbreviation length are limited to a certain extent. For machine learning-based methods, the decision is made by the machine-learning algorithm. For pair classification, proper features are designed to learn the abbreviations and expansions. Typical works include those on SVM (Support Vector Machine)-based [16], HMM (Hidden Markov Model)-based [17], CRF (Conditional Random Field)-based [18], and latent-state neural CRF-based [19] methods. Henriksson et al. [20] stated that different types of texts contain different features and that different models can extract different features. Based on this idea, they suggested a mixed multi-model and multi-corpus model to judge the final full forms based on candidates. The main purposes of above methods are to select best expansion for the abbreviation, the main differences are in the following three aspects: type of abbreviation, the search rule and search scope. the details of these three aspects for these methods are listed in Table 1.

Highly accurate abbreviation dictionaries have been applied widely in biomedical big data. However, there are some drawbacks. Firstly, it is time-consuming and laborious to maintain dictionaries, and secondly, dictionaries are not suitable for ad hoc abbreviations.

2.2. ASD

For standard abbreviations, the main task of ASD is to choose a suitable expansion from the dictionary according to the context. The machine learning is the most used methods for ASD. In [21–24], a classifier for each abbreviation is trained according to the context. In [21], Moon et al. presented evaluations of three kinds of classifier (naïve Bayes, SVM, decision trees) in terms of their features, context window sizes, orientations, and minimum training sample sizes and proposed the best configuration of these parameters. In [22], Wu et al. described the use of word embedding to construct the context. Two new SBE (surrounding-based embedding) context modes called LR_SBE (left–right surrounding based embedding) feature and MAX_SBE (maximum surrounding based embedding) feature are integrated as the context. Then, the context is entered into the SVM to select the proper sense. In [23], the authors discussed the use of two kinds of context mode (SBE and term frequency–inverse document frequency-based embedding modes) to model the context, followed by abbreviation disambiguation by calculating the cosine similarity to choose the most similar one from the given candidates. Hua et al. [24] established a profile for each sense of an abbreviation from the discharge summaries and admission notes. During disambiguation, the cosine similarities between the context vector of the abbreviation and the profile vectors are calculated, and the sense corresponding to the highest similarity score is selected as the correct one.

In contrast to the techniques employed in the abovementioned studies, deep learning methods [25,26] have the obvious advantage that feature engineering can be avoided. In [26], Ahmed et al. suggested a deep learning model to train a vector for the context of each sense of each abbreviation. For disambiguation, the cosine-similarities for the context vector of the sentence with the context vector of each sense are calculated, and the sense with the maximum value is selected. In [25], Joopudi et al. proposed a convolutional neural network with one convolutional kernel, a max-pooling layer, and a fully connected feed-forward neural network layer followed by a fully connected softmax classifier. Except for the embedding of surrounding words, the location and part of speech information of the word are considered in the context.

2.3. Other related application areas

The processing methods for abbreviations are different in different research fields. The handling details of abbreviations in some typical fields are described below.

Text Normalization The target of text normalization is to convert the informal text into standard formal form. It is a critical step in the variety of tasks involving speech and language technologies. In [27], Zhang et al. test the effect of automatic normalization on dependency parsing by using automatically derived parse trees of normalized sentences as reference. It is shown that

Table 1
Key detail of the methods.

References	Type of abbreviation	Search rule	Search scope
[14]	Acronym	LCS of initial	Any subsequence in ten words before the acronym
[15]	Acronym	LCS of initial, heuristics rules	Any subsequence in the sentence
[16]	Acronym	SVM	Any sequences in the sentence exceed the number of characters in the acronym in length
[17]	Acronym	LCS and the same initials	$\min(\|A\|+5, \|A\|*2)$
[18]	Acronym	CRF	Any subsequence in the sentence
[19]	Acronym	LNCRF	All the subsequence in the document
[20]	Abbreviation	Semantic distance of context	All the words in the text

the performance of the normalizer is directly tied to the performance of a downstream dependency parser. In [28], Wu Y et al. presented a framework called CARD(clinical abbreviation recognition and disambiguation) to handle the abbreviation in clinical data that leverages previously developed methods, including: (1) machine learning based approaches to recognize abbreviations from a clinical corpus, (2) clustering-based semi-automated methods to generate possible senses of abbreviations, and (3) profile-based word sense disambiguation methods for clinical abbreviations. In [29], Yonghui Wu et al. propose a hybrid strategy that combines a machine learning based method using SVM, a profile-based method using Vector Space Model, and a majority-sense method to resolve ambiguous abbreviations appeared at different frequency levels. In [30], Pierre Zweigenbaum et al. propose a supervised abbreviation resolution system. The system learns a MaxEnt(Maximum Entropy) model from the training data, based on a simple feature set that combines UMLS(Unified Medical Language System) knowledge with information gathered from the EMR text.

Software maintenance: Information retrieval techniques are being exploited by an increasing number of tools supporting software maintenance activities. Identifiers are among the most valuable sources of information for software maintenance. When a programmer introduces an abbreviation (e.g., rect) as an identifier, the difficulty of understanding the identifier is increased. Corazza et al. [31] proposed a method of automatically splitting identifiers into their composing words and expanding abbreviations. The solution is based on a graph model and performs linearly in time with respect to the dictionary size. Because only complete words are considered in the identifier, multi-word abbreviations cannot be handled effectively, but this approach regarding identifiers is the most commonly used. Alatawi et al. [32] proposed a Bayesian unigram-based inference to expand abbreviations automatically into their original words to enhance source code maintenance. In this technique, a list of candidate words is extracted automatically from the source code for a given abbreviation and the statistical properties of the unigram of the abbreviation are employed as evidence to find the best candidate word. Due to the use of the unigram model, this approach cannot be utilized to address abbreviations that are expandable into phrases. Alatawi et al. [12] presented a bigram-based approach that could be used to expand an abbreviation into a phrase automatically with multiple unigrams. In this method, the abbreviation is firstly divided into segments. Then, the candidates for each segment are generated, where the candidates consist of the sequences of full forms of segment. Finally, the best phrase is chosen from the phrase candidates according to the bigram language model (LM).

Schema matching: Abbreviations, particularly ad hoc abbreviations, are used in schema more frequently because of the limitation of the length of attribute's name. Ratinov and Gudes [33] proposed a method in which a neural network is utilized to judge the relation between two elements (one is an abbreviation, and the other is a full form). Firstly, the relation between the abbreviation and full form is formalized into a sequence containing four operations, and a neural network is then trained to judge the correctness of the operation sequence. In this method, the relation between the abbreviation and full form is judged by the neural network, but the candidate full form must be prepared in advance, which is not the case in all fields. Sorrentino et al. [34] used four resources to evaluate each expansion and selected the expansion with the maximum evaluation value to normalize the abbreviation.

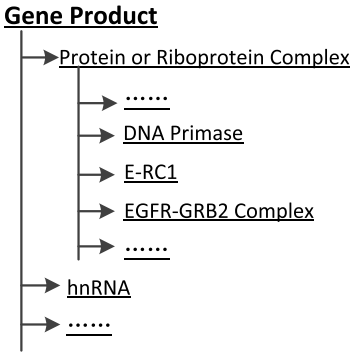


Fig. 2. A XML schema.

In addition to the abovementioned areas, some other fields are influenced by abbreviations, such as data integration [35], speech recognition [36] and string join [37,38] etc. In these fields, abbreviation dictionaries play an important role in abbreviation operation.

3. Problem analysis and definition

In order to describe the problem and the proposed scheme clearly, the symbols used in the paper are shown in Table 2.

Example 1. Naked eDNA, most of it released by cell death, is nearly ubiquitous in the environment.

In Example 1, the abbreviation Edna cannot be found in many acronym dictionaries. By analyzing the abbreviations, it could be seen that the abbreviation consist of an initial of word "extracellular" and a standard abbreviation "DNA". Therefore, even for these standard abbreviations, the dictionary could not provide their full forms.

Example 2. Fig. 2 shows a part of class structure of NCIt (National Cancer Institute thesaurus). To understand the semantic information of the schema correctly, it is necessary to obtain the semantics of each element in the schema. But in this schema, the designer gives some abbreviations like "EGFR-GRB2 (Epidermal Growth Factor Receptor- Growth Factor Receptor Bound Protein 2)" which does not included in the dictionary. In fact, it is composed of two abbreviations "EGFR" and "GRB2".

Table 2
Symbols definition.

Symbol	Definition	Symbols	Definition
∂	Partition of an abbreviation	$CANDSS(S)$	Candidate semantic set of S
∂^i	i-th block of a partition	$SS(S)$	Semantic set of S
$PreES(\partial^i)$	Set of all words with prefix	$CLUSS(S)$	Semantic set of a cluster of S
$ParES\left(\partial_j\right)$	Set of expansions for partition ∂_j	$Rationality(\partial)$	Rationality value of partition
$PrefixAbbr(Ph)$	Prefix abbreviation of phrase Ph	ρ(w)	Ratio of the words that can appear adjacent to word w in all word sets.
$PARTISET(S)$	Partition set of abbreviation S	$Sim\ (phrase_1, phrase_2)$	Similarity of $phrase_1$ and $phrase_2$
$\|partition(S)\|$	Number of elements in $PARTISET(S)$	$E\text{-}PARTISET(S)$	Effective partition set of abbreviation S

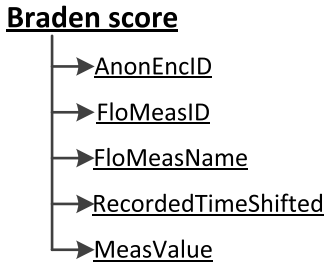


Fig. 3. Database schema.

Example 3. Fig. 3 presents a database schema. To understand the semantic information in a schema correctly, it is necessary to obtain the semantics of each element it contains. However, in this schema, the designer has used some ad hoc abbreviations, such as AnonEncID, FloMeasID, FloMeasName and MeasValue, according to his personal preferences. Obviously, the full forms of these abbreviations would not be included in an abbreviation dictionary, because they are only used by the designer rather than widely accepted.

The examples provided above demonstrate that abbreviations are always changing. Thus, the dictionary-based method is ineffective, especially for ad hoc abbreviations. After analyzing the characteristics of the abovementioned abbreviations, we found that the problem becomes easy if the correct partition of the abbreviation in question is obtained. For example, if WEBGL is divided into "WEB" and "GL", the real semantic can be found easily by using a dictionary. Based on this idea, this paper proposes an LM-based automatic abbreviation expansion method to enumerate all possible full forms of abbreviations.

The concepts related to abbreviation expansion can be described as follows.

Definition 1. Character set: $\Phi = \{26$ English letters$|$ a, b, c… , z$\}$.

Definition 2. Ordered string: The arbitrary and repeatable characters in the character set constitute ordered string S, which is denoted as $S = \{\alpha_1, \alpha_2, \alpha_3, \ldots\ldots \alpha_n\}$ (n is the length of the ordered string, denoted as $LEN(S)$).

Definition 3. Word: If S is given a specific meaning, S is defined as word w.

Definition 4. Dictionary: The set of all words, denoted as D.

Definition 5. Phrase: An ordered sequence comprised of several words is defined as a phrase and denoted as $Ph = (w_1 w_{2\ldots\ldots} w_{n-1} w_n)$ (n is the length of the phrase, denoted as $LEN(Ph)$).

Definition 6. Prefix abbreviation: For any phrase, the prefix abbreviation consists of the first few characters of each word in sequence and is denoted as $S = PrefixAbbr(Ph)$.

Definition 7. Abbreviation expansion: For any abbreviation S, list all phrases satisfying the prefix abbreviation rule for the specified D and LM.

Algorithm 1: LMAAE-METHOD

Require: S: the abbreviation, D: dictionary contained all words, LM: a language model.
Output: CLUSS(S): all the expansion of S;
1: function LMAAE-METHOD (S, D, LM)
2: PARTISET(S) ← Enumerate all the partitions of S.
3: E-PARTISET(S) ← the partition in PARTISET(S) with rationality() > threshold
4: for in E-PARTISET(S)
5: for ∂^i in
6: PreES(∂^i) ← all the items with prefix ∂^i in D
7: end for
8: CANDSS(S) ← CANDSS(S) ∪ $\prod_{i=1}^{len(\partial)}$ PreES(∂^i)
9: end for
10: SS(S) ← all the items e in CANDSS(S) with LM(e) > threshold
11: CLUSS(S) ← cluster all the items in SS(S)

The automatic abbreviation expansion procedure is shown in Algorithm 1. Algorithm 1 takes S, D, and the LM as inputs. Firstly, the prefix abbreviation is partitioned and the partitions are evaluated and sorted according to certain rules. Then, each partition is restored according to D to obtain all possible candidate expansions. Next, the candidate semantic set is evaluated and filtered according to the LM to obtain the semantic set. Finally, by clustering the semantic set, the clustering semantic set is obtained as the expansion result. The details of this algorithm are provided in Section 4.

4. Automatic prefix abbreviation expansion

4.1. Element name decomposition

Block Arbitrary continuous characters constitute a block of abbreviation S. For instance, block $\alpha_{i\to j} = \left\{\alpha_i, \ldots\ldots \alpha_j\right\} (1 \le i \le j \le n)$ is a block of S called block∂^i.

Coverage: Partition S into m blocks $\{\partial^1, \partial^2, \ldots\ldots, \partial^m\}$ and each character is contained in at least one block. The set of these blocks is called a coverage of S.

Partition: For a coverage $\partial = \left\{\partial^1, \partial^2, \ldots\ldots, \partial^m\right\}$, if each character α_i of S belongs to only one block, is called a partition of S, and m is the number of blocks in this partition, denoted as $m = LEN(\partial)$.

Reasonable block: For a block ∂^i of partition , if there exist words with prefixes ∂^i in D, then ∂^i is designated as a reasonable block; otherwise, it is designated as an unreasonable block.

Reasonable partition: For a partition $\partial = \{\partial^1, \partial^2, \ldots\ldots, \partial^m\}$ of S, if all of the blocks in this partition are reasonable blocks, this partition is considered to be reasonable; otherwise, it is unreasonable.

Partition set: All reasonable partitions of S constitute the partition set of S, denoted as $PARTISET(S)$.

For any S, the partition set can be obtained by employing Algorithm 2.

Algorithm 2: Segmentation

```
Require: S: the abbreviation, start: start position of
           partition, end: end position of partition,
         n: total number of blocks, m: number of blocks
           partitioned, partitions: the m blocks
           partitioned
Output: PARTISET(S): all the partitions of S with n blocks;
 1:  function Segmentation (S, start, end, n, m, partitions)
 2:    if m + 1 == n then
 3:      partitions ← partitions ∪ S[strart, end]
 4:      PARTISET(S) ← PARTISET(S) ∪ partitions
 5:    eif end-start+1==n-m then
 6:      for c in [end, start]
 7:        partitions ← partitions ∪ S[c]
 8:      PARTISET(S) ← PARTISET(S) ∪ partitions
 9:    else
10:      for c in [start, end-(n-m+1)]
11:        partitions ← partitions ∪ S[start, c]
12:        Segmentation(S, c+1, end, n+1, m, partitions)
13:      end for
14:  end if
```

All partitions of S with length N can be listed by using Algorithm 2. Therefore, by modifying the value of N, $PARTISET(S)$ can be listed. The following is the analysis of the number of partitions in $PARTISET(S)$.

$P(LEN(S), i)$: The number of partitions with length i of S with length $LEN(S)$.

$|PARTISET(S)|$: The number of the partitions of S.

The relationship between the two abovementioned numbers is

$$|partition(S)| = \sum_{i=1}^{LEN(S)} P(LEN(S), i). \tag{1}$$

Here, for each i, the value of $P(LEN(S), i)$ is as follows:

$$P\left(LEN\left(S\right), 1\right) = 1 \tag{2}$$

$$P(LEN(S), 2) = \sum_{i=1}^{LEN(S)-1} P\left(LEN(S) - i, 1\right) = LEN\left(S\right) - 1 = C_{LEN(S)-1}^{1}; \tag{3}$$

$$P(LEN(S), 3) = \sum_{i=1}^{LEN(S)-2} P\left(LEN(S) - i, 2\right) = \frac{1}{2} * \left(LEN\left(S\right) - 2\right)\left(LEN\left(S\right) - 1\right) = C_{LEN(S)-1}^{2} \tag{4}$$

$$P\left(LEN\left(S\right), LEN\left(S\right) - 1\right) = \sum_{i=1}^{2} P\left(LEN\left(S\right) - i, LEN\left(S\right) - 2\right) = LEN\left(S\right) - 1 = C_{LEN(S)-1}^{LEN(S)-2} \tag{5}$$

$$P(LEN(S), LEN(S)) = \sum_{i=1}^{1} P\left(LEN(S) - i, LEN(S) - 1\right) = 1. \tag{6}$$

. . . . . .

The sum of the formulas above is

$$|partition(S)| = \sum_{i=0}^{LEN(S)-1} C_{LEN(S)-1}^{i} = 2^{LEN(S)-1}. \tag{7}$$

According to Eq. (7), the number of partitions of S is $2^{LEN(S)-1}$, so the worst-case time complexity of a partition is $O(2^{LEN(S)-1})$. Usually, this time complexity is known as unsolvable. However, in this situation, length restrictions can be imposed on the abbreviation and partition to obtain the upper limit of the number of partitions.

The upper partition length limit corresponds to the number of words in the full forms. Intuitively, an upper limit N must exist. Fig. 4 presents the statistical results for the full forms (of which there are about 900,000) on a well-known abbreviation website.[3] Fig. 4 demonstrates that when the number of words exceeds 7, there are fewer corresponding phrases, but when the number of words is greater than or equal to 8, about 99.5% of the full forms can be included. In a practical situation, by setting the upper partition length limit to 8, the complexity of the algorithm can be decreased considerably while negligibly affecting the precision.

Let the upper limit of words be N. Then, the number of partitions is given by

$$\begin{cases} |partition\left(S\right)| = \sum_{i=0}^{LEN(S)-1} C_{LEN(S)-1}^{i} = 2^{LEN(S)-1} (N \geq LEN(S)) \\ |partition\left(S\right)| = \sum_{i=0}^{N-1} C_{LEN(S)-1}^{i} < 2^{LEN(S)-1} \left(N < LEN\left(S\right)\right). \end{cases} \tag{8}$$

According to Eq. (8), when the length of S exceeds a certain threshold, the complexity is decreased from exponential magnitude to polynomial magnitude.

Based on the concept of reasonable partition, if and only if each block is a reasonable block, the partition is a reasonable partition. Therefore, in the segmentation process, the recursion can be pruned when an unreasonable block is encountered. Doing so can reduce the complexity, whose quantitative analysis is presented below.

There must be a correct partition corresponding to the expansion when segmenting any prefix abbreviation. Except for the blocks of the correct partition, those of the other partitions are distributed randomly across the character sequence space. Before presenting the quantitative analysis of the ratio of reasonable blocks in the random character sequence space, it is necessary to introduce the following concepts.

Trie tree: A complete tree with a degree of 26. The 26 children of each node correspond sequentially to the 26 letters.

Word node: For each node in a trie tree, if the string from the root to the node corresponds to a word in the dictionary, the word is called a word node.

Prefix node: All of the nodes in the path from the root to the word node are called prefix nodes.

Prefix number: For each node, the total number of word nodes in its subtree is denoted as the prefix number of the node.

Non-prefix node: A node is a non-prefix node when its prefix number is zero.

When dividing a character sequence, all of the blocks in the partition besides the correct one are randomly distributed in the dictionary tree. Assuming that the length of a random block is 5, the corresponding prefix number is $26^5 = 11,881,376$. However, the number of the words in the dictionary is just 1,193,517, and the number of reasonable blocks with length 5 must be less than this value[4]. Consequently, only 10% of all prefixes with length 5 are reasonable. The actual statistics for the dictionary are shown in Fig. 5.

According to Fig. 5, the ratio of reasonable prefixes is about 14% in a random letter sequence when the prefix length is 4

[3] www.abbreviations.com.

[4] The dictionary is generated from a 14 Gb corpus text from twitter, the number of the words is 552345.

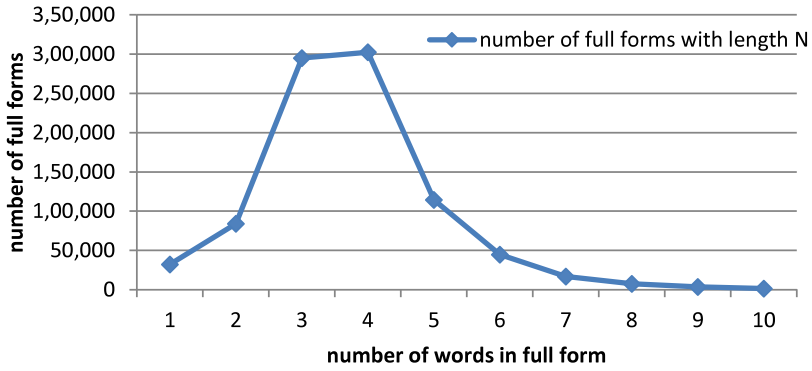


Fig. 4. Statistical results for the full forms.

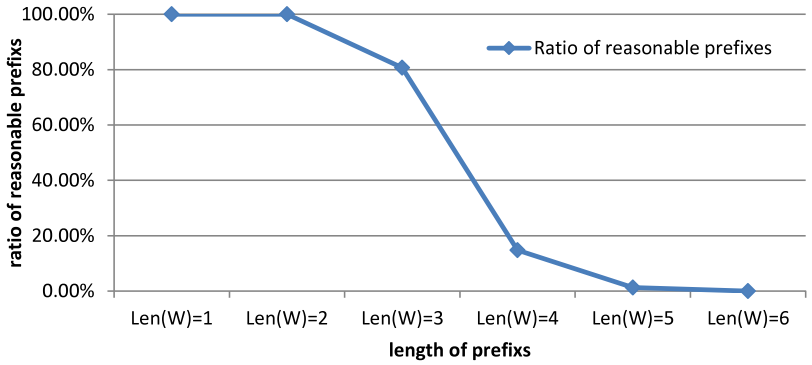


Fig. 5. Ratios of reasonable prefixes with different lengths.

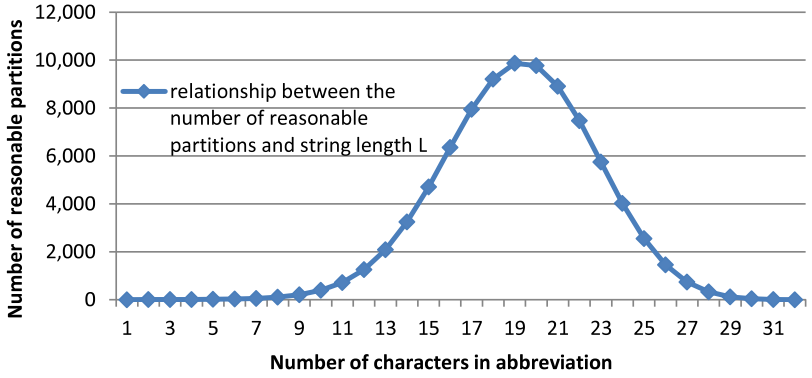


Fig. 6. Number of reasonable partitions in the string with length *L*.

and 1% when the length is 5. Most of the random blocks with lengths of more than 5 are not reasonable. To analyze the number of reasonable partitions, we set the prefix length to 4 (it was assumed that a prefix was unreasonable when its length was more than 4; otherwise, it was considered to be reasonable). The number of reasonable partitions in this case is shown in Fig. 6.

In Fig. 6, the number of reasonable partitions increases slowly with increasing abbreviation length. When the length is 8, the number of reasonable partitions is less than 100. The number of reasonable partitions reaches its maximum of 9,867 when the length is 19, then drops rapidly with further increasing *L*. For abbreviations with lengths greater than 32, there is only one reasonable partition, the correct one.

According to the analysis presented above, after pruning in advance based on the reasonability of blocks, the complexity is greatly reduced from the original polynomial magnitude to a constant magnitude, which is acceptable in practical situations.

4.2. Partition evaluation

For a given abbreviation, all reasonable partitions can be obtained by applying the algorithm in Section 4.1. When a user omits some characters in a phrase to form an abbreviation, some rules are always followed, so the partitions can be evaluated according to the rules. The next three rules are selected as the evaluation criterion for partitions.

Accordingly, in the LMAAE method, Eq. (9) is utilized to calculate the rationality of each reasonable partition except the initial partition, denoted as $Rationality(\partial)$. The rationality of initial partition is set to a fixed value.

$$Rationality(\partial) = \alpha/LEN(\partial) + \beta * \left(\frac{\sum_{i=1}^{m} \left(length\left(\partial^i\right) * Completeness\left(\partial^i\right) \right)}{\sum_{i=1}^{m} length\left(\partial^i\right)} \right) \times (\alpha + \beta = 1). \quad (9)$$

Table 3
Expansion table.

	Meas	Value
Expansion set	Measles	Value
	Measurable	Valued
	Measurably	Valuer
	Measure	Values
	.....	.....

Value of *Completeness* $\left(\partial^i\right)$ could be calculated using Eq. (10)

$$Completeness\left(\partial^i\right) = \begin{cases} 1 & \text{if } length\left(\partial^i\right) > 3 \\ 0 & \text{if } length\left(\partial^i\right) \le 3 \end{cases} \tag{10}$$

According to Eq. (9), the rationality of each reasonable partition can be calculated. Then, the Max-N strategy is used to filter some partitions. A partition that is not filtered is an effective partition, and all of the effective partitions form an effective partition set.

4.3. Partition expansion

For the effective partitions selected by using the Max-N strategy, it would be simple to enumerate all possible expansions by enumerating all possible expansions for each block and then determining their Cartesian product. However, the result is too large to handle. This section presents an analysis of the complexity of the expansion and suggests an optimized strategy based on statistical LMs. Theoretical analysis of the strategy is provided, demonstrating that this method can limit the result set to an acceptable range. Related definitions are listed below.

Prefix expansion set *PreES*(∂^i): For any prefix ∂^i, all of the words in D with prefix ∂^i make up the expansion set of ∂^i, denoted as $PreES(\partial^i) = \left\{w_1^{\partial^i}, w_2^{\partial^i}, \ldots\ldots, w_n^{\partial^i}\right\}$. In $PreES(\partial^i)$, $w_j^{\partial^i}$ is the word with prefix ∂^i, and n is its length, denoted as $LEN(PreES(\partial^i))$. In a trie tree, $LEN(PreES(\partial^i))$ is the prefix number of ∂^i.

Partition expansion set *ParES* $\left(\partial_j\right)$: For any partition ∂_j, the Cartesian product of all of the $PreES(\partial_j^i)$ forms the partition expansion set $ParES\left(\partial_j\right)$, denoted as shown in Eq. (11). In $ParES\left(\partial_j\right)$, each expansion is a sequence of words denoted as $WS = (w_{i_1}^{\partial^1}, w_{i_2}^{\partial^2}, \ldots\ldots, w_{i_m}^{\partial^m})$ $(i_j \le LEN\left(PreES\left(\partial_j^j\right)\right), 1 \le j \le LEN(\partial_j))$.

$$ParES\left(\partial_j\right) = PreES\left(\partial_j^1\right) \times PreES\left(\partial_j^2\right) \ldots\ldots \times PreES\left(\partial_j^m\right) (m = LEN\left(\partial_j\right)) \tag{11}$$

Candidate semantic set *CandSS*(S): For prefix abbreviation S, the intersection of all of the $ParES\ (\partial_i)$ makes up the candidate semantic set of S, denoted as $CandSS\ (S) = \bigcup_{i=1}^{LEN(PARTISET(S))} Pares(\partial_i)$.

For the partition ∂_j = meas, value of "measvalue", the possible expansions of all of the blocks are shown in Table 3.

The Cartesian product is determined for the set in each column to obtain the partition expansion set of ∂_j, $ParES\left(\partial_j\right)$ = {measles value, measurable value,.....}, measure value, measure valued, ...}. The number of elements in $ParES\left(\partial_j\right)$ could be calculated by multiplying the lengths of all of the $PreES(\partial_j^i)$ as follows:

$$LEN(ParES\left(\partial_j\right)) = \prod_{i=1}^{i \le LEN(\partial_j)} LEN(PreES(\partial_j^i)). \tag{12}$$

For each reasonable block ∂_j^i, $LEN(PreES(\partial_j^i))$ is the prefix number of the corresponding node of ∂_j^i in the trie tree. The average number of prefixes with length n can be estimated based on the statistics of the average number of nodes on the n-th floor of the trie tree. The real data are shown in Table 4.

Table 4
Statistical analysis of average number of prefixes.

Number of characters	1	2	3	4	5
Average number of prefixes	38,461	1479	60	3	1.4

By analyzing the problem and considering the data in Table 4, the length of $PreES(\partial_j^i)$ could be replaced by the average value $\frac{|D|}{26^{LEN(\partial_j^i)}}$. If ∂_j is a partition of an abbreviation of length L and $LEN\left(\partial_j\right) = m$, then 26^{4m-L} is the estimated value of $LEN(ParES\left(\partial_j'\right))$. This value is too large to handle effectively. To solve this problem, this paper suggests a strategy based on statistical LMs.

Based on the example in Table 3, most of the expansions of S in $CandSS(S)$ are meaningless phrases. A statistical LM [39,40] has the function of distinguishing meaningless phrases from the candidate expansions. In the LMAAE method, each candidate is evaluated using the LM and the candidates with evaluation values under a threshold are filtered.

Co-occurrence rate: The ratio of the words that can appear adjacent to word w in all word sets, denoted as $\rho(w)$, is given by

$$\rho(w) = \frac{|\text{set of words next to w}|}{|\text{set of all words}|}. \tag{13}$$

Because every word has a different co-occurrence rate, the co-occurrence rate of every word is set equal to the average co-occurrence rate in the dictionary for the convenience of analysis. By employing the co-occurrence rate, $LEN(ParES\left(\partial_j\right))$ can be calculated as follows:

$$LEN\left(ParES\left(\partial_j\right)\right) = 26^{4m-L} * \rho(w)^{m-1}. \tag{14}$$

In Wikipedia data, the average co-occurrence rate is 0.00046. Assuming $\rho = 0.0005$, $LEN(S) = 10$, when the number of blocks M is 5, and the number of effective expansions is about 9 according to Eq. (14). All of the effective expansions of each effective partition constitute the semantic set of S (denoted as $SS(S)$).

4.4. Semantic set clustering

Because a root can yield numerous words with the same prefix and similar semantics, there are many phrases with close similar semantics in the set, which would disturb subsequent operations, such as ASD. A clustering method is introduced below to merge phrases with similar semantics.

For any two phrases ($phrase_1 = \{w_1^1, w_1^2, \ldots\ldots, w_1^n\}$, $phrase_2 = \{w_2^1, w_2^2, \ldots\ldots, w_2^m\}$), the semantic similarity is defined as

$$\text{Sim}\ (phrase_1, phrase_2) = EM(\sum_{i=1}^{n} vector\left(w_1^i\right), \sum_{i=1}^{m} vector\left(w_2^i\right)). \tag{15}$$

In Eq. (15), $vector\left(w_1^i\right)$ denotes the word embedding of w_1^i, and EM is the Euclidean distance between two vectors. Compared with sentences, there are fewer words and the semantics are simpler in phrases. Therefore, this method can provide better results by taking the average value of the word vectors as the vector representation of a phrase.

The mean-shift clustering algorithm is selected to cluster phrases, as shown in Algorithm 3. In Algorithm 3, the expansion

phrases in $SS(S)$ are divided into several categories according to semantic distance. In each class, the phrase with the maximum statistical probability is selected as the semantic representation of that category, which consists of the clustering semantic set ($CLUSS(S)$) of S.

Algorithm 3: Phrases_clustering

```
Require: nodes: all the expansions in SS(S), bandwidth: a
         threshold for classification.
Output: CLUSS(S): all the expansion of S;
1:   function Phrases_clustering (nodes,bandwidth)
2:     for nd1 in nodes
3:       if nd1 is not classified
4:         N ← nd1
5:         for nd2 in nodes
6:           if EM(nd2,N)<bandwidth
7:             M ← M ∪ nd2
8:             P(M,nd2)+=1
9:           endif
10:        end for
11:        shift ← 0
12:        do
13:          N ← N+shift
14:          for nd3 in M
15:            shift ← shift+nd3-N
16:          end for
17:        while(shift > threshold1) //threshold1 is the
     convergence condition
18:        for C in CLUSS(S)
19:          if EM (N, central point of C)<threshold2
20:            For nd4 in M
21:              C ← C ∪ nd4
22:              P(C,nd4)+=1
23:            end for
24:          endif
25:        endfor
26:        if no C in CLUSS(S) meet the condition EM (N,
     central point of C)<threshold2
27:          CLUSS(S) ← CLUSS(S) ∪ M
28:        endif
29:      end if
30:      for C in CLUSS(S)
31:        C ← the node with max statistical language
     models(SLM) value
32:      endfor
33:    endfor
```

5. Experiments

The LMAAE method can obtain all of the candidate expansions automatically. It can effectively handle ad hoc abbreviations, which the dictionary-based method cannot. For standard abbreviations, it can supplement the dictionary-based method when abbreviations are missing from the dictionary. To validate the effectiveness of the LMAAE method, the following experiments were conducted.

1. In order to validate the precision of LMAAE method to standard abbreviation, a simulation experiment is conducted on the abbreviation in the dictionary. The result shows that more than 80% of the expansions in the dictionary are listed by LMAAE.

2. In order to validate the precision of LMAAE method to ad hoc abbreviation, a set of ad hoc abbreviation is selected from OAEI[5] (Ontology Alignment Evaluation Initiative). The precision of Top-20 reached 75%.

3. In order to validate the effectiveness of LMAAE to the related fields, two simulation experiments are conduct on schema matching and text normalization. The result shows that LMAAE could increase the precision exceed 5 percent in related fields.

5.1. Comparison with the dictionary for standard abbreviations

In some research fields such as ASD, users must choose the most suitable semantics from all of the candidate semantics, so the completeness of the set of candidate semantics is crucial. To validate the completeness of the LMAAE method, the following experiment was performed.

Test data consisting of 1000 abbreviations and all of their full forms were extracted from an online abbreviation dictionary (www.abbreviations.com). Then, the LMAAE method was used to obtain the expansions of the abbreviations. The precision and recall given by Eqs. (16) and (17), respectively, were used to evaluate the results.

Precision The proportion of the full forms in the abbreviation dictionary that are listed by the automatic algorithm.

$$\text{Precision} = \frac{|AUTOFF(Abbr) \cap DICFF(Abbr)|}{|DICFF(Abbr)|} \tag{16}$$

Recall: The proportion of the full forms listed by the automatic algorithm that are in the abbreviation dictionary.

$$\text{Recall (Abbr)} = \frac{|AUTOFF(Abbr) \cap DICFF(Abbr)|}{|AUTOFF(Abbr)|} \tag{17}$$

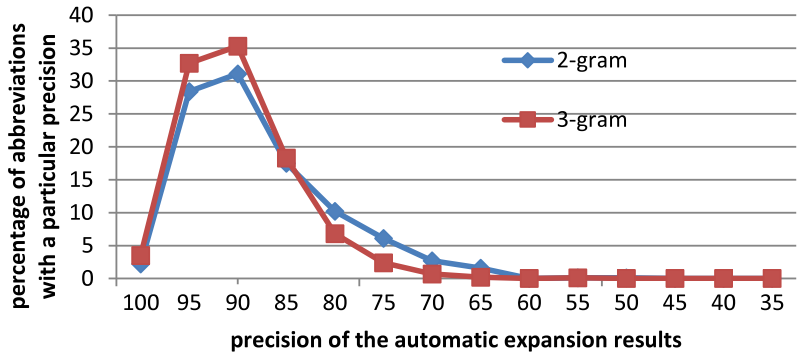
In Eqs. (16) and (17), $AUTOFF(Abbr)$ is the set of all full forms listed by the automatic algorithm, and $DICFF(Abbr)$ is the set of all full forms in the abbreviation dictionary. The experimental results are presented in Fig. 7.

Fig. 7(a) shows the precision of the LMAAE method. The horizontal axis corresponds to the precision of the automatic expansion results, and the vertical axis shows the percentage of abbreviations with a particular precision. The blue and red curves were obtained using the 2- and 3-gram LMs, respectively. For instance, on the 3-gram curve, when the vertical value is 35.3 and the horizontal value is 90, it means that 35.3% of the abbreviations have automatic expansion results with precisions between 90% and 95%. When the 2-gram LM is used, 2.2% of the expansion results include all of the full forms in the dictionary, and 89.4% of the expansion results have precisions greater than 80%. When the 3-gram LM is used, 3.5% of the expansion results include all of the full forms in the dictionary, and 96.6% of the expansion results have precisions greater than 85%. The data in Fig. 6(a) demonstrate that most of the full forms in the dictionary could be obtained by the automatic algorithm.

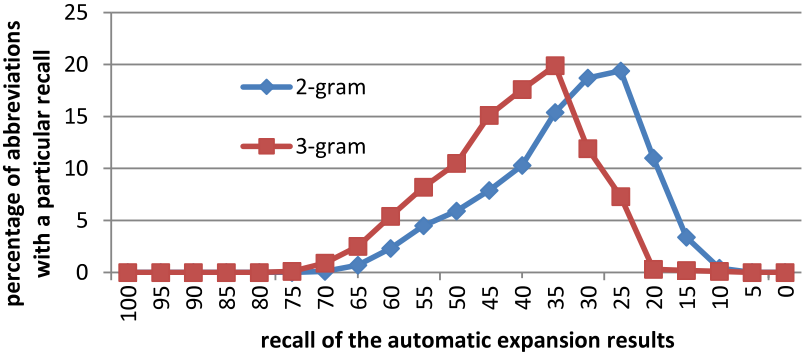
Fig. 7(b) depicts the recall of the LMAAE method. The horizontal axis corresponds to the recall of the automatic expansion results, and the vertical axis shows the percentage of the abbreviations with a particular recall. The meanings of the curves are similar to those of the precision curves. When the 2-gram LM is used, only 4.1% of the expansion results have recalls greater than 60%, and most of the results have recalls around 30%. When the 3-gram LM is used, only 8.9% of the expansion results have recalls greater than 60%, and most of the results have recalls around 35%. The data in Fig. 7(b) indicate that most of the expansions generated by the algorithm are not listed in the abbreviation dictionary.

According to the data in Fig. 7(a) and 7(b), the automatic expansion results include most of the full forms in the dictionary but about 65% of the expansion result are not listed in the dictionary(called Non-dictionary expansion). Non-dictionary expansion is not the error, but the correct expansion abbreviated seldom. Because not any dictionary could contain all the abbreviation and its expansion, so even the standard abbreviation,

[5] http://oaei.ontologymatching.org/.



(a). Precision of the LMAAE method.



(b). Recall of the LMAAE method.

Fig. 7. Precision and recall of LMAAE method.

Table 5
Accuracy of expansions for ad hoc abbreviations.

	TOP1	TOP3	TOP5	TOP10	TOP20
2-g	40.2%	58.7%	63.1%	68.8%	73.2%
3-g	42.5%	60.3%	68.3%	72.1%	76.9%

dictionary-based method could not handle all of them. When the corresponding expansion is not contained in the dictionary, the automatic expansion set would give an effective supplement to increase the precision of ASD.

5.2. Expansion of ad hoc abbreviations

Ad hoc abbreviations appear in schemas, knowledge graphs etc., and generally are not included in abbreviation dictionaries. To validate the effectiveness of the LMAAE method at expanding ad hoc abbreviations, 531 ad hoc abbreviations were extracted from the data set used in the OAEI contest, and the correct full forms were generated manually (only 73 full forms could be found in the abbreviation dictionary). The experimental results are summarized in Table 5.

Through the data in Table 5, LMAAE shows good performance for ad hoc abbreviations. For 3-gram model, its TOP1 precision is 42.5%(for 42.5% of the abbreviation generate expansion by LMAAE, the first expansion is correct) and TOP3(the correct expansion is in the first three expansion) precision reaches 60.3%. When the number of candidate is expands to 20, it achieves 76.9% precision. For 2-gram model, the result is slightly low than 3-gram model. The data shows that LMAAE could expand ad hoc abbreviation effectively.

5.3. Schema matching improvement

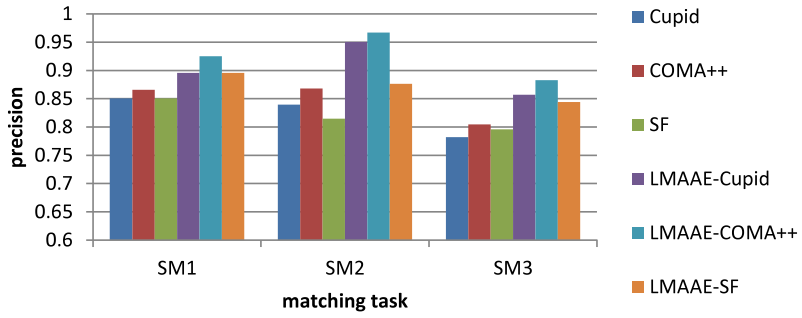
The objective of schema matching is to find the correspondences between the elements of the schema. Abbreviations in the names of the elements in a schema is a significant obstacle in schema matching. In this section, we describe the use of the LMAAE method to handle the abbreviations in the schemas to verify its effectiveness in this application. Three schema matching tasks were conducted, as shown in Table 6, and three classical schema-matching algorithms (Cupid [41], COMA++ [42], and SF (similarity flooding) [43]) were selected.

In this experiment, the three algorithms were firstly used to obtain the matching results directly (the abbreviations were handled by a standard abbreviation dictionary such as WordNet). Then, the LMAAE method was used to preprocess the abbreviations in the schemas. For each abbreviation, the LMAAE method generated its top 10 full forms. When calculating the name similarity between elements, the maximum name similarity was selected as the final value. Standard measurements including precision (defined as the ratio between the number of correctly detected result and the total number of abbreviations), recall(defined as the ratio between the number of results correctly detected by the system and the total number of abbreviations), and F1 score (calculated as 2*precision*recall/(precision+recall)) were used to evaluate the matching results. The experimental results are presented in Fig. 8.

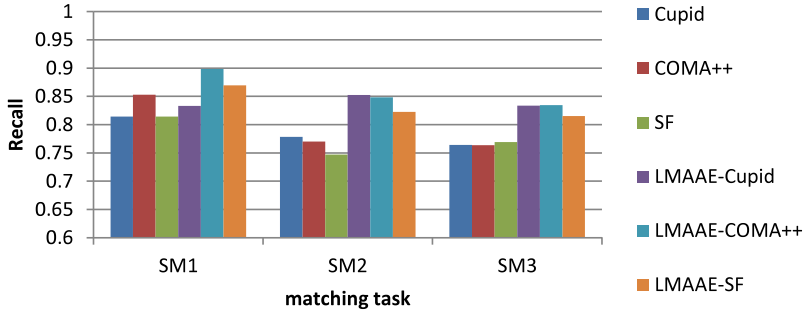
The data in Fig. 8 demonstrate that the LMAAE method can improve the performance effectively. The LMAAE method provides a precision 7% greater than that of the classical algorithm on average. The improvement is different for each matching task: 5% for SM1, 9% for SM2, and 7% for SM3 and the maximum improvement

Table 6
Overview of the schemas.

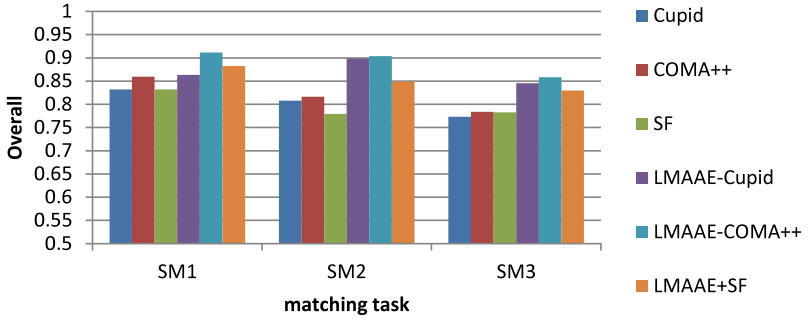
Task	Schema	Relation	Attribute	Abbreviation	Remark
SM1	DBS1 DBT1	12 14	73 75	7 5	Databases of two forums
SM2	DBS2 DBT2	43 38	307 294	32 36	Buying-stock-selling database of two dealers
SM3	DBS3 DBT3	126 133	905 938	54 92	Enterprise resource planning (ERP) database of two manufacturing firms in the same domain



(a) Precision.



(b) Recall.



(c) F1-Score.

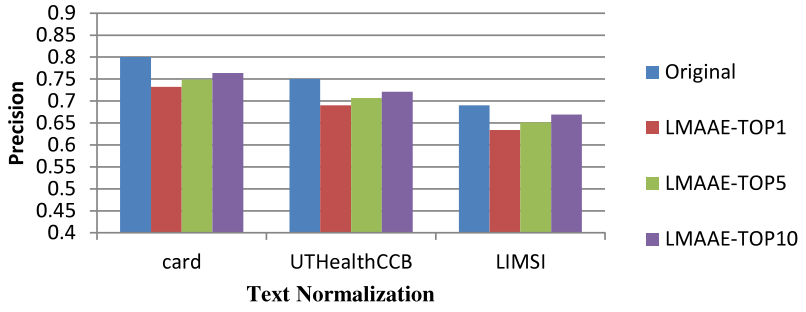
Fig. 8. Precision, recall and F1-score of six matching method for different task.

is 84%–95% for SM2 with Cupid. For recall, the LMAAE method exhibits 6% improvement over the classical algorithm on average. The improvement is 4% for SM1, 7% for SM2, and 6% for SM3, and the maximum improvement is 74.7%–82.2% for SM2 with SF. The F1-Score is increased by 6% on average with the LMAAE method compared to the classical algorithm. The improvement 4.5% for SM1, 8% for SM2, and 6.5% for SM3, and the maximum improvement is 81%–90% for SM2 with COMA++.

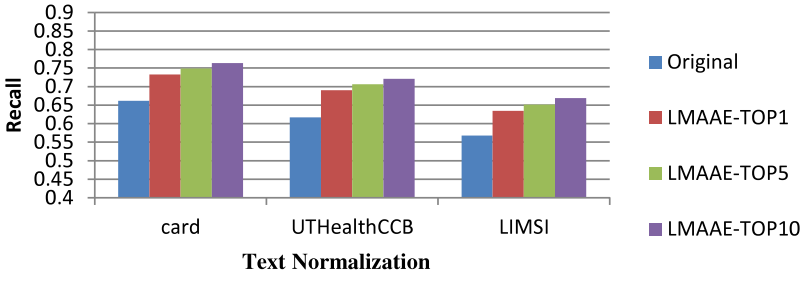
Thus, the precision and overall score are obviously improved by using the LMAAE method before the classical schema matching algorithm. The improvements are especially notable for SM2, which included more abbreviations. The LMAAE method could enhance the schema matching effectively. To highlight the effects for the elements with abbreviations, Table 7 provides a statistical overview of the abbreviation matching results. In Table 7, the numbers in the cells represent the number of abbreviation matches in each task for different matching methods. For example, the data in the second row means that: there are 5 matching relations in ground-truth with attribute in element's name, Cupid and COMA++ generate 3 of them, SF generate 2 of them. But with the assist of LMAAE, Cupid and COMA++ generate all of them, SF generate 4 of them.

Table 7
Statistics of the abbreviation matching results.

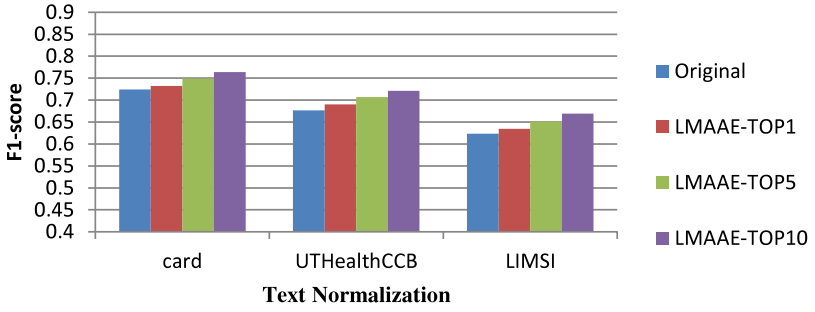
Task	Ground-truth	Cupid	COMA++	SF	LMAAE-Cupid	LMAAE-COMA++	LMAAE+SF
SM1	5	3	3	2	5	5	4
SM2	40	17	18	17	34	36	32
SM3	85	26	28	23	62	65	60



(a) Precision.



(b) Recall.



(c) F1-Score.

Fig. 9. Precision, recall and F1-score of Text Normalization.

As demonstrated by Table 7, with the help of the LMAAE method, the classical algorithms could identify more correct matches for the abbreviations. For SM2, only 17 abbreviation matches were identified with WordNet, but with the LMAAE method, 34 relations were identified. For SM3, the precision was improved considerably. Thus, the LMAAE method could solve the abbreviation problem effectively.

5.4. Text normalization improvement

For further verification of the effective of LMAAE, a hot research area text normalization is select in this section. One of the main targets of text normalization is to get the correct expansion of the abbreviation in the text. For this experiment, we leveraged the ShARe (Shared Annotated Resources) corpus, a subset of de-identified discharge summary, electrocardiogram, echocardiogram, and radiology reports from about 30,000 ICU (Intensive Care Unit) patients provided by the MIMIC (Multi-parameter Intelligent Monitoring in Intensive Care) [44]. In this experiment, a data set with 80 clinical texts and 3024 abbreviations is selected from the corpus and three recent works (CARD [28], UTHealthCCB [29], LIMSI [30]) are used to normalize the text. The same measurements as Section 5.3 are used to evaluate the result. The experimental results are presented in Fig. 9.

The data in Fig. 9 show the result about three original methods and the original method assisted by the LMAAE to handle the abbreviation not included in dictionary(called non-dictionary abbreviation). The method LMAAE-TOP1 means that the LMAAE generate only one expansion for non-dictionary abbreviation. For the other two methods, LMAAE generate 5 or 10 expansions respectively. The result demonstrates that the LMAAE method

can improve the overall performance of text normalization. But the improvement is different for each evaluation indicator. For precision, because all three kinds of original methods do not generate expansion for non-dictionary abbreviation, so the precision is decreased when LMAAE generate expansion for non-dictionary abbreviation. The maximal decline is generated by CARD method, the precision is decrease from 80% of original to 73% of LMAAE-TOP1. But for recall, LMAAE could increase the performance greatly because the number of standard result is fixed. The maximal rise is generated by UTHealthCCB method , the recall is increased from 61% of original to 72% of LMAAE-TOP10. For the comprehensive indicator F1-score, LMAAE could increase the value for each method especially for UTHealthCCB method from 67% of original to 72% of LMAAE-TOP10. In summary, LMAAE could improve performance of the method for text normalization.

6. Conclusions and future work

This paper proposed an LM-based automatic prefix abbreviation reduction method called the LMAAE method. With the widespread use of abbreviations, the dictionary-based method that is currently used to address the abbreviation problem has considerable limitations, especially for ad hoc abbreviations. By analyzing the common rules for abbreviating, it was determined that, compared with reductions, abbreviations conceal word division information (words are linked together) and the complete forms of words (only the prefix is extracted). Therefore, automatic abbreviation reduction can be used to restore latent information. In the proposed automatic reduction technique, the division information is firstly restored by dividing the abbreviations. Secondly, the complete forms are obtained by restoring each block. Finally, according to the LM, the reductions are filtered and clustered to obtain the final results. The time complexity of this algorithm is higher than the dictionary-based method. By analyzing the time complexity, combined with practical problems, an optimized partition algorithm was developed, with the time complexity decreased from exponential to constant. Simultaneously, the time complexity of the reduction algorithm was decreased considerably. The experimental results additionally demonstrated that the proposed method has higher comprehensiveness for general abbreviations. In other words, the reductions can cover most dictionary results. The proposed method also has higher accuracy for ad hoc abbreviations. So it is a good choice to complement for the dictionary-based method.

Acknowledgments

This work was supported by the Fundamental Research Funds for the Central Universities, China (No. CZY18018, CZP19004), the National Science Foundation of China (No. 61772562), the Hubei Provincial Natural Science Foundation of China for Distinguished Young Scholars (No. 2017CFA043), the Key Project of Hubei Provincial Science and Technology Innovation Foundation of China (No.2018ABB1485), and the Youth Elite Project of State Ethnic Affairs Commission of China.

References

[1] S.G. Alonso, I. de la Torre Díez, J.J.P.C. Rodrigues, S. Hamrioui, M. López-Coronado, J.P. Papa, A.X. Falcao, V.H.C. de Albuquerque, J.M.R.S. Tavares, A systematic review of techniques and sources of big data in the healthcare sector, J. Med. Syst. 41 (2017) 183.

[2] J.P. Papa, A.X. Falcao, V.H.C. de Albuquerque, J.M.R.S. Tavares, Efficient supervised optimum-path forest classification for large datasets, Pattern Recognit. 45 (2012) 512–520.

[3] R.M. Al abdi, A.E. Alhitary, E.W.A. Hay, A.K. Al-bashir, Objective detection of chronic stress using physiological parameters, Med. Biol. Eng. Comput. (2018) 1–14.

[4] X. Zhai, Z. Li, K. Gao, Y. Huang, L. Lin, L. Wang, Research status and trend analysis of global biomedical text mining studies in recent 10 years, Scientometrics 105 (2015) 509–523.

[5] K. Kreimeyer, M. Foster, A. Pandey, N. Arya, G. Halford, S.F. Jones, R. Forshee, M. Walderhaug, T. Botsis, Natural language processing systems for capturing and standardizing unstructured clinical information: A systematic review, J. Biomed. Inform. 73 (2017) 14–29.

[6] E. Jiménez-Ruiz, C. Meilicke, B.C. Grau, I. Horrocks, Evaluating mapping repair systems with large biomedical ontologies, Descr. Logics. (2013) 246–257.

[7] W. Ammar, K. Darwish, A. El Kahki, K. Hafez, ICE-TEA: in-context expansion and translation of English abbreviations, in: CICLing'11 Proc. 12th Int. Conf. Comput. Linguist. Intell. Text Process. - vol. Part II, 2011: pp. 41–54.

[8] M. Oleynik, M. Kreuzthaler, S. Schulz, Unsupervised abbreviation expansion in clinical narratives, MedInfo. (2017) 539–543.

[9] J. Politis, S. Lau, J. Yeoh, C. Brand, D. Russell, D. Liew, Overview of shorthand medical glossary (OMG) study., Intern. Med. J. 45 (2015) 423–427.

[10] I. Harrow, E. Jiménez-Ruiz, A. Splendiani, M. Romacker, P. Woollard, S. Markel, Y. Alam-Faruque, M. Koch, J. Malone, A. Waaler, Matching disease and phenotype ontologies in the ontology alignment evaluation initiative, J. Biomed. Semant. 8 (2017) 55.

[11] E. Chondrogiannis, V. Andronikou, T. Varvarigou, E. Karanastasis, Semantically-enabled context-aware abbreviations expansion in the clinical domain, in: Proc. 9th Int. Conf. Bioinforma. Biomed. Technol. 2017: pp. 89–96.

[12] A. Alatawi, W. Xu, J. Yan, The expansion of source code abbreviations using a language model. in: 2018 IEEE 42nd Annu. Comput. Softw. Appl. Conf. 2018: pp. 370–375.

[13] T.L. Stedman, Stedman's medical abbreviations, Acron. Symb. (2012).

[14] K. Taghva, J. Gilbreth, Recognizing acronyms and their definitions, Int. J. Doc. Anal. Recognit. 1 (1999) 191–198.

[15] S. Yeates, Automatic extraction of acronyms from text. in: New Zeal. Comput. Sci. Res. Students' Conf. (1999) pp. 117–124.

[16] J. Xu, Y. Huang, Using SVM to extract acronyms from text, Soft Comput. 11 (2006) 369–373.

[17] U. Hahn, P. Daumke, S. Schulz, K.G. Markó, Cross-language mining for acronyms and their completions from the web, Discov. Sci. (2005) 113–123.

[18] J. Liu, J. Chen, Y. Zhang, Y. Huang, Learning conditional random fields with latent sparse features for acronym expansion finding, in: Proc. 20th ACM Int. Conf. Inf. Knowl. Manag. 2011: pp. 867–872.

[19] J. Liu, C. Liu, Y. Huang, Multi-granularity sequence labeling model for acronym expansion identification, Inf. Sci. (Ny). 378 (2017) 462–474.

[20] A. Henriksson, H. Moen, M. Skeppstedt, V. Daudaravičius, M. Duneld, Synonym extraction and abbreviation expansion with ensembles of semantic spaces, J. Biomed. Semant. 5 (2014) 6.

[21] S. Moon, S.V.S. Pakhomov, G.B. Melton, Automated disambiguation of acronyms and abbreviations in clinical texts: window and training size considerations. in: AMIA Annu. Symp. Proc. 2012: pp. 1310–1319.

[22] Y. Wu, J. Xu, Y. Zhang, H. Xu, Clinical abbreviation disambiguation using neural word embeddings, in: Proc. BioNLP 15, 2015: pp. 171–176.

[23] C. Li, L. Ji, J. Yan, Acronym disambiguation using word embedding, in: AAAI'15 Proc. Twenty-Ninth AAAI Conf. Artif. Intell. 2015: pp. 4178–4179.

[24] H. Xu, P.D. Stetson, C. Friedman, Combining corpus-derived sense profiles with estimated frequency information to disambiguate clinical abbreviations. in: AMIA Annu. Symp. Proc. 2012: pp. 1004–1013.

[25] V. Joopudi, B. Dandala, M. Devarakonda, A convolutional route to abbreviation disambiguation in clinical text, J. Biomed. Inform. 86 (2018) 71–78.

[26] A.G. Ahmed, M.F.A. Hady, E. Nabil, A. Badr, A language modeling approach for acronym expansion disambiguation, in: Int. Conf. Intell. Text Process. Comput. Linguist. 2015: pp. 264–278.

[27] C. Zhang, T. Baldwin, H. Ho, B. Kimelfeld, Y. Li, Adaptive Parser-Centric Text Normalization, 2013, pp. 1159–1168.

[28] Y. Wu, J.C. Denny, S.T. Rosenbloom, R.A. Miller, D.A. Giuse, L. Wang, C. Blanquicett, E. Soysal, J. Xu, H. Xu, A long journey to short abbreviations: developing an open-source framework for clinical abbreviation recognition and disambiguation (CARD), J. Am. Med. Inform. Assoc. 24 (2016).

[29] Y. Wu, B. Tang, M. Jiang, S. Moon, J.C. Denny, H. Xu, Clinical acronym/abbreviation normalization using a hybrid approach, Unkn. J 1179 (2013).

[30] P. Zweigenbaum, L. Deléger, T. Lavergne, A. Névéol, A. Bodnari, A Supervised Abbreviation Resolution System for Medical Text, CLEF (Working Notes), 2013.

[31] A. Corazza, S. Di Martino, V. Maggio, LINSEN: An efficient approach to split identifiers and expand abbreviations, in: 2012 28th IEEE Int. Conf. Softw. Maint. 2012: pp. 233–242..

[32] A. Alatawi, W. Xu, D. Xu, Bayesian unigram-based inference for expanding abbreviations in source code, in: 2017 IEEE 29th Int. Conf. Tools with Artif. Intell. 2017: pp. 543–550..

[33] L. Ratinov, E. Gudes, Abbreviation expansion in schema matching and web integration, in: Proc. 2004 IEEE/WIC/ACM Int. Conf. Web Intell. 2004: pp. 485–489..
[34] S. Sorrentino, S. Bergamaschi, M. Gawinecki, L. Po, Schema label normalization for improving schema matching, Data Knowl. Eng. 69 (2010) 1254–1273.
[35] J.J. Helly, T.T. Elvins, D. Sutton, D. Martinez, A method for interoperable digital libraries and data repositories, Futur. Gener. Comput. Syst. 16 (1999) 21–28.
[36] A. Mansikkaniemi, M. Kurimo, Adaptation of morph-based speech recognition for foreign names and acronyms, IEEE Trans. Audio Speech Lang. Process. 23 (2015) 941–950.
[37] W. Tao, D. Deng, M. Stonebraker, Approximate string joins with abbreviations, Very Larg. Data Bases 11 (2017) 53–65.
[38] I. Spasic, Acronyms as an integral part of multi-word term recognition – A token of appreciation, IEEE Access. 6 (2018) 8351–8363.
[39] P. Wang, Y. Qian, F.K. Soong, L. He, H. Zhao, Learning distributed word representations for bidirectional LSTM recurrent neural network, in: Proc. 2016 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. 2016: pp. 527–533..
[40] M.S. Kudinov, A.A. Romanenko, A hybrid language model based on a recurrent neural network and probabilistic topic modeling, Pattern Recognit. Image Anal. 26 (2016) 587–592.
[41] J. Madhavan, P.A. Bernstein, E. Rahm, Generic schema matching with cupid, Very Larg. Data Bases. (2001) 49–58.
[42] D. Aumueller, H.H. Do, S. Massmann, E. Rahm, Schema and ontology matching with COMA++, in: Proc. ACM SIGMOD Int. Conf. Manag. Data, 2005: pp. 906–908.
[43] S. Melnik, H. Garcia-Molina, E. Rahm, Similarity flooding: a versatile graph matching algorithm and its application to schema matching, in: Proc. 18th Int. Conf. Data Eng. 2002: pp. 117–128.
[44] M. Saeed, C. Lieu, G. Raber, R.G. Mark, MIMIC II: a massive temporal ICU patient database to support research in intelligent patient monitoring, Comput. Cardiol. (2002) 641–644.

Xiaokun Du is currently an lecturer at the College of Computer Science of South-Central University for Nationalities, China. He received her Ph.D. degree from Huazhong University of Science and Technology (HUST), China, in 2010. Her research interests include data integration, text mining and sentiment analysis.

Rongbo Zhu is currently a Professor in College of Computer Science of South-Central University for Nationalities, China, and he is the director of Institute of Smart Cities. He received the B.S. and M.S. degrees in Electronic and Information Engineering from Wuhan University of Technology, China, in 2000 and 2003, respectively; and Ph.D. degree in communication and information systems from Shanghai Jiao Tong University, China, in 2006. From August 2011 to August 2012, he was a research scholar in CNSR group at Virginia Tech, USA. Dr. Zhu has published over 70 papers in international journals and conferences in the areas of mobile computing and cognitive wireless networks. He is an Associate Editor IEEE Access, International Journal of Radio Frequency Identification Technology and Applications, and Lead Guest Editor of 7 international journals.

Yanhong Li is currently an associate professor at the College of Computer Science of South-Central University for Nationalities, China. She received her Ph.D. degree from Huazhong University of Science and Technology (HUST), China, in 2011. Her research interests include spatial information and communication, and multimedia network technology.

Ashiq Anjum is currently a professor of Distributed Systems at the University of Derby, UK. His research interests include data intensive distributed systems, block chain, Internet of Things and high performance analytics platforms. Currently he is investigating high performance distributed platforms to efficiently process video and genomics data. Dr. Anjum has more than 70 international academic publications. He has been part of the EC funded projects in distributed systems and large scale analytics such as Health-e-Child (IP, FP6), neuGrid (STREP, FP7) and TRANSFORM (IP, FP7). He has been a general chair, programmer chair, organizing chair, track chair, publicity chair and workshop chair for more than 20 international conferences. Prof. Anjum has been a member of the technical program committees for more than 40 international conferences. He is the member of British Computer Society, IEEE, ACE, and SIGHPC. He is the Fellow of Higher Education Academy and the champion of European Grid Infrastructure.