Sqn2Vec: Learning Sequence Representation via Sequential Patterns with a Gap Constraint

Dang Nguyen[1(✉)], Wei Luo[1], Tu Dinh Nguyen[1], Svetha Venkatesh[1], and Dinh Phung[2]

[1] Center for Pattern Recognition and Data Analytics, School of Information Technology, Deakin University, Geelong, Australia
{d.nguyen,wei.luo,tu.nguyen,svetha.venkatesh}@deakin.edu.au
[2] Faculty of Information Technology, Monash University, Clayton Campus, Melbourne, VIC 3800, Australia
dinh.phung@monash.edu

Abstract. When learning sequence representations, traditional pattern-based methods often suffer from the data sparsity and high-dimensionality problems while recent neural embedding methods often fail on sequential datasets with a small vocabulary. To address these disadvantages, we propose an *unsupervised* method (named **Sqn2Vec**) which first leverages *sequential patterns* (SPs) to increase the vocabulary size and then learns *low-dimensional continuous* vectors for sequences via a neural embedding model. Moreover, our method enforces a *gap constraint* among symbols in sequences to obtain meaningful and discriminative SPs. Consequently, **Sqn2Vec** produces significantly better sequence representations than a comprehensive list of state-of-the-art baselines, particularly on sequential datasets with a relatively small vocabulary. We demonstrate the superior performance of **Sqn2Vec** in several machine learning tasks including sequence classification, clustering, and visualization.

1 Introduction

Many real-world applications such as web mining, text mining, bio-informatics, system diagnosis, and action recognition have to deal with sequential data. The core task of such applications is to apply machine learning methods, for example, K-means or Support Vector Machine (SVM) to sequential data to find insightful patterns or build effective predictive models. However, this task is challenging since machine learning methods typically require inputs as fixed-length vectors, which are not applicable to sequences.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-10928-8_34) contains supplementary material, which is available to authorized users.

© Springer Nature Switzerland AG 2019
M. Berlingerio et al. (Eds.): ECML PKDD 2018, LNAI 11052, pp. 569–584, 2019.
https://doi.org/10.1007/978-3-030-10928-8_34

A well-known solution in data mining is to use *sequential patterns* (SPs) as features [6]. This approach first mines SPs from the dataset, and then represents each sequence in the dataset as a feature vector with binary components indicating whether this sequence contains a particular sequential pattern. We can see the dimension of the feature space is huge since the number of SPs is often large. Consequently, this leads to the high-dimensionality and data sparsity problems.

To reduce the dimension of the feature space, many researchers have tried to extract only *interesting SPs* under an *unsupervised* setting [5,9,17] or *discriminative SPs* under a *supervised* setting [3,6,19]. The methods discover interesting SPs, e.g., closed SPs [17], compressing SPs [9], and relevant SPs [5] without using the sequence labels. Although these methods can reduce the number of generated patterns, thus solving the high-dimensionality problem, they still suffer from the data sparsity problem. The methods discover discriminative SPs using different measures, e.g., information gain [6], support-cohesion [19], and behavioral constraint [3], which involve the sequence labels. Although these methods often show good performances in sequence classification, they usually require the labels for all training examples, which is often unrealistic in many real applications.

Recently, neural embedding approaches have been introduced to learn *low-dimensional continuous* embedding vectors for sequences using neural networks in a fully *unsupervised* manner. These methods primarily focus on text, where they learn embedding vectors for documents [2,10] and show significant improvements over non-embedding methods, e.g., Bag-of-Words, in several applications such as document classification and sentiment analysis. However, they have two limitations. First, they mostly learn embedding vectors based on atoms in data (i.e., words), but do not consider sets of atoms (i.e., phrases). Second, they often perform poorly on datasets with a relatively small vocabulary [8]. In our experiments, the performances of document embedding methods dramatically reduce on sequential datasets whose the vocabulary size is less than 300.

Our Approach. To overcome the disadvantages of traditional pattern-based methods and recent embedding methods, we propose a novel *unsupervised* method (named **Sqn2Vec**) for learning sequence embeddings. In particular, we first extract a set of SPs which satisfy a gap constraint from the dataset. We then adapt a document embedding model to learn a vector for each sequence by predicting not only its belonging symbols but its SPs as well. By doing this, we can learn *low-dimensional continuous* vectors for sequences, which solves the weakness of pattern-based methods. We also take into account sets of atoms (i.e., SPs) during the learning process, which solves the weakness of embedding methods. More importantly, by considering both singleton symbols and SPs, we can increase the vocabulary size, which results in our better embeddings on sequential datasets with a small vocabulary. Moreover, since **Sqn2Vec** is fully unsupervised, it can be directly used for learning sequence embeddings in domains where labeled examples are difficult to obtain and the learned representations are well-generalized to different tasks such as sequence classification, clustering, and visualization.

To summarize, we make the following contributions:

1. We propose **Sqn2Vec**, an *unsupervised* embedding method, for learning *low-dimensional continuous* feature vectors for sequences.
2. We propose two models in **Sqn2Vec**, which *learn* sequence embeddings by predicting its belonging singleton symbols and SPs. The learned embeddings are meaningful and discriminative.
3. We demonstrate **Sqn2Vec** in both sequence classification and sequence clustering tasks, where it significantly outperforms the state-of-the-art baselines on 10 real-world sequential datasets.

2 Related Work

2.1 Sequential Pattern Based Methods for Sequence Representation

SPs have been widely used to construct feature vectors for sequences [6], which are essential inputs for different machine learning tasks. However, using SPs as features often suffers from the data sparsity and high-dimensionality problems. Recent SP-based methods have tried to extract only interesting or discriminative SPs. Several approaches have been proposed for mining interesting SPs. For example, Lam et al. [9] discovered compressing SPs which can optimally compress the dataset w.r.t an encoding scheme. In [5], a probabilistic approach was developed to mine relevant SPs which are able to reconstruct the dataset. Although interesting SPs can help to reduce the number of generated patterns (i.e., the feature space), they still suffer from the data sparsity problem.

To discover discriminative SPs, existing approaches have used the sequence labels during the mining process. For example, they use the label information to compute information gain [6], support-cohesion [19], or behavioral constraint [3]. Although discriminative SPs are useful for classification, they require sequence labels, making the mining process *supervised*. Related to sequence classification, SPs have been also used to build a set of predictive rules for classification, often called *sequential classification rules*. These rules represent the strong associations between SPs and labels, which can be used directly for prediction (i.e., they are used as rule-based classifiers) [19] or indirectly for prediction (i.e., they are used as features in other classifiers) [4].

2.2 Embedding Methods for Sequence Representation

Most existing approaches for sequence embedding learning mainly focus on text, where they learn embedding vectors for documents [2,10]. These methods have shown impressive successes in many natural language processing tasks such as document classification and sentiment analysis. They, however, are not suitable for sequential data in bio-informatics, navigation systems, and action recognition since different from text, these sequential datasets have a very small vocabulary size (i.e., the small number of distinct symbols). For example, the human DNA sequences only consist of four nucleotides A, C, G, and T. Related to sequence classification, several deep neural network models (also called *supervised embedding methods*) have been introduced such as long short term memory (LSTM)

networks and bidirectional LSTM (Bi-LSTM) networks [16]. Since these methods require labels, their embeddings are not general enough to effectively apply to unsupervised tasks such as sequence clustering.

As far as we know, learning embedding vectors for sequences based on SPs has not been studied yet. In this paper, we propose the first approach which utilizes information of both singleton symbols and SPs to learn sequence embeddings. Different from discriminative pattern-based methods and supervised embedding methods, our method is fully *unsupervised*. Moreover, our method leverages SPs to capture the sequential relations among symbols as SP-based methods while it learns dense representations as embedding methods.

3 Framework

3.1 Problem Definition

Given a set of symbols $\mathcal{I} = \{e_1, e_2, ..., e_M\}$, a *sequential dataset* $\mathcal{D} = \{S_1, S_2, ..., S_N\}$ is a set of sequences where each sequence S_i is an ordered list of symbols [18]. The symbol at the position j in S_i is denoted as $S_i[j]$ and $S_i[j] \in \mathcal{I}$.

Our goal is to learn a mapping function $f : \mathcal{D} \rightarrow \mathbb{R}^d$ such that every sequence $S_i \in \mathcal{D}$ is mapped to a d-dimensional continuous vector. The mapping needs to capture the similarity among the sequences in $\mathcal{D}$, in the sense that S_i and S_j are similar if $f(S_i)$ and $f(S_j)$ are close to each other on the vector space, and vice versa. The matrix $\mathbf{X} = [f(S_1), f(S_2), ..., f(S_N)]$ then contains feature vectors of sequences, which can be direct inputs for many traditional machine learning and data mining tasks, particularly classification and clustering.

3.2 Learning Sequence Embeddings Based on Sequential Patterns

To learn sequence embeddings, one direct solution is to apply document embedding models [2,10] to the sequential dataset, where each sequence is treated as a document and symbols are treated as words. However, as we discussed in Sect. 1, existing document embedding methods are not suitable for sequential datasets in bio-informatics or system diagnosis since these datasets have a relatively small vocabulary (i.e., the very small number of symbols).

To improve the performances of document embedding models on such kind of sequential data, we propose to learn sequence embeddings based on SPs instead of singleton symbols. By doing this, we can increase the vocabulary size since the number of SPs is much larger than the number of symbols.

Sequential Pattern Discovery. Following the notations in [18], we define a sequential pattern as follows. Let $\mathcal{I} = \{e_1, e_2, ..., e_M\}$ be a set of symbols and $\mathcal{D} = \{S_1, S_2, ..., S_N\}$ be a sequential dataset.

Definition 1 ***(Subsequence).*** *Given two sequences* $S_1 = \{e_1, e_2, ..., e_n\}$ *and* $S_2 = \{e'_1, e'_2, ..., e'_m\}$, S_1 *is said to be a subsequence of* S_2 *or* S_1 *is contained in*

S_2 (denoted $S_1 \subseteq S_2$), if there exists a one-to-one mapping $\phi : [1, n] \rightarrow [1, m]$, such that $S_1[i] = S_2[\phi(i)]$ and for any positions i, j in S_1, $i < j \Rightarrow \phi(i) < \phi(j)$. In other words, each position in S_1 is mapped to a position in S_2, and the order of symbols is preserved.

Definition 2 ***(Subsequence occurrence).*** *Given a sequence $S = \{e'_1, e'_2, ..., e'_m\}$ and a subsequence $X = \{e_1, e_2, ..., e_n\}$ of S, a sequence of positions $o = \{i_1, ..., i_m\}$ is an occurrence of X in S if $1 \leq i_k \leq m$ and $X[k] = S[i_k]$ for each $1 \leq k \leq n$, and $i_k < i_{k+1}$ for each $1 \leq k < n$.*

Example 1. $X = \{g, t\}$ (or $X = gt$ for short) is a subsequence of $S = gaagt$. There are two occurrences of X in S, namely $o_1 = \{1, 5\}$ and $o_2 = \{4, 5\}$.

Definition 3 ***(Subsequence support).*** *Given a sequential dataset $\mathcal{D}$, the support of a subsequence X is defined as $sup(X) = \frac{|\{S_i \in \mathcal{D} | X \subseteq S_i\}|}{|\mathcal{D}|}$, i.e., the fraction of sequences in $\mathcal{D}$, which contain X.*

Definition 4 ***(Sequential pattern).*** *Given a minimum support threshold $\delta \in [0, 1]$, a subsequence X is said to be a sequential pattern if $sup(X) \geq \delta$.*

A sequential pattern can capture the sequential relation among symbols, but it does not pay attention on the gap among its elements. In bio-data and text data, this gap is very important because SPs whose symbols are far away from each other are often less meaningful than those whose symbols are close in the sequences. For example, consider a text dataset with two sentences S_1 = "machine learning is a field of computer science" and S_2 = "machine learning gives computer systems the ability to learn". Although two SPs X_1 = {machine, learning} and X_2 = {machine, computer} are found in both S_1 and S_2, X_2 is less meaningful than X_1 due to the large gap between "machine" and "computer". In other words, the two words "machine" and "computer" are in two different contexts. We believe that if we restrict the distance between two neighboring elements in a sequential pattern, then this pattern is more meaningful and discriminative. We define a sequential pattern satisfying a gap constraint as follows.

Definition 5 ***(Gap constraint and satisfaction).*** *A gap is a positive integer, $\triangle > 0$. Given a sequence $S = \{e'_1, e'_2, ..., e'_m\}$ and an occurrence $o = \{i_1, ..., i_m\}$ of a subsequence X of S, if $i_{k+1} \leq i_k + \triangle$ $(\forall i_k \in [1, m-1])$, then we say that o satisfies the $\triangle$-gap constraint. If there is at least one occurrence of X satisfies the $\triangle$-gap constraint, we say that X satisfies the $\triangle$-gap constraint.*

Example 2. Among two occurrences of $X = gt$ in $S = gaagt$, namely $o_1 = \{1, 5\}$ and $o_2 = \{4, 5\}$, only o_2 satisfies the 1-gap constraint (i.e., $\triangle = 1$) since $5 \leq 4 + \triangle$. We say that X satisfies the 1-gap constraint because at least one of its occurrences does.

Definition 6 ***(Sequential pattern satisfying a $\triangle$-gap constraint).*** *Given a sequential dataset $\mathcal{D}$, a gap constraint $\triangle > 0$, and a minimum support threshold*

$\delta \in [0, 1]$, the support of a subsequence X in $\mathcal{D}$ with the $\triangle$-gap constraint, denoted $sup(X, \triangle)$, is the fraction of sequences in $\mathcal{D}$, where X appears as a subsequence satisfying the $\triangle$-gap constraint. X is called a sequential pattern which satisfies the $\triangle$-gap constraint if $sup(X, \triangle) \geq \delta$.

Note that we consider the subsequences with length 1 (i.e., they contain only one symbol) satisfy any $\triangle$-gap constraint. Hereafter, we call a subsequence X a sequential pattern with the meaning that X is a sequential pattern satisfying a $\triangle$-gap constraint.

Example 3. Let consider an example sequential dataset as shown in Fig. 1(a). Assume that $\triangle = 1$ and $\delta = 0.7$. The subsequence $X = ag$ is contained in three sequences S_1, S_2, and S_4, and it also satisfies the 1-gap constraint in these three sequences. Thus, its support is $\text{sup}(X, \triangle) = 3/4 = 0.75$. We say that $X = ag$ is a sequential pattern since $\text{sup}(X, \triangle) \geq \delta$. With $\triangle = 1$ and $\delta = 0.7$, there are in total five SPs discovered from the dataset, as shown in Fig. 1(b), and each sequence now can be represented by a set of SPs, as shown in Fig. 1(c).

Seq	Symbols
S_1	{c, a, g, a, a, g, t}
S_2	{t, g, a, c, a, g}
S_3	{g, a, a, t}
S_4	{a, g}

(a)

SP	Symbols	*sup*
X_1	{a}	1.00
X_2	{g}	1.00
X_3	{t}	0.75
X_4	{a, g}	0.75
X_5	{g, a}	0.75

(b)

Seq	SPs
S_1	$\{X_1, X_2, X_3, X_4, X_5\}$
S_2	$\{X_1, X_2, X_3, X_4, X_5\}$
S_3	$\{X_1, X_2, X_3, X_5\}$
S_4	$\{X_1, X_2, X_4\}$

(c)

Fig. 1. Two forms of a sequence: a set of single symbols and a set of SPs. Table (a) shows a sequential dataset with four sequences where each of them is a set of symbols. Table (b) shows five SPs discovered from the dataset (here, $\triangle = 1$ and $\delta = 0.7$). Table (c) shows each sequence represented by a set of SPs.

Sequence Embedding Learning. After associating each sequence with a set of SPs, we follow the Paragraph Vector-Distributed Bag-of-Words (PV-DBOW) model introduced in [10] to learn embedding vectors for sequences. Given a target sequence S_t whose representation needs to be learned, and a set of SPs $\mathcal{F}(S_t) = \{X_1, X_2, ..., X_l\}$ contained in S_t, our goal is to maximize the log probability of predicting the SPs $X_1, X_2, ..., X_l$ which appear in S_t:

$$\max \sum_{i=1}^{l} \log \Pr(X_i \mid S_t) \tag{1}$$

Furthermore, $\Pr(X_i \mid S_t)$ is defined by a softmax function:

$$\Pr(X_i \mid S_t) = \frac{\exp(g(X_i) \cdot f(S_t))}{\sum_{X_j \in \mathcal{F}(\mathcal{D})} \exp(g(X_j) \cdot f(S_t))}, \tag{2}$$

where $g(X_i) \in \mathbb{R}^d$ and $f(S_t) \in \mathbb{R}^d$ are the embedding vectors of the sequential pattern $X_i \in \mathcal{F}(S_t)$ and the sequence S_t respectively, and $\mathcal{F}(\mathcal{D})$ is the set of all SPs discovered from the dataset $\mathcal{D}$.

Calculating the summation $\sum_{X_j \in \mathcal{F}(\mathcal{D})} \exp(g(X_j) \cdot f(S_t))$ in Eq. 2 is very expensive since the number of SPs in $\mathcal{F}(\mathcal{D})$ is often very large. To solve this problem, we approximate it using the negative sampling technique [13]. The idea is that instead of iterating over all SPs in $\mathcal{F}(\mathcal{D})$, we randomly select a relatively small number of SPs which are not contained in the target sequence S_t (these SPs are called *negative SPs*). We then attempt to distinguish the SPs contained in S_t from the negative SPs by minimizing the following binary objective function of logistic regression:

$$\mathcal{O}_1 = -\left[\log \sigma(g(X_i) \cdot f(S_t)) + \sum_{n=1}^{K} \mathbb{E}_{X^n \sim \mathcal{P}(X)} \log \sigma(-g(X^n) \cdot f(S_t))\right], \quad (3)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is a sigmoid function, $\mathcal{P}(X)$ is the set of negative SPs, X^n is a negative sequential pattern draw from $\mathcal{P}(X)$ for K times, and $g(X^n) \in \mathbb{R}^d$ is the embedding vector of X^n.

We minimize $\mathcal{O}_1$ in Eq. 3 using stochastic gradient descent (SGD) where the gradients are derived as follows:

$$\begin{aligned} \frac{\partial \mathcal{O}_1}{\partial g(X^n)} &= -\sigma(g(X^n) \cdot f(S_t) - \mathbb{I}_{X_i}[X^n]) \cdot f(S_t) \\ \frac{\partial \mathcal{O}_1}{\partial f(S_t)} &= -\sum_{n=0}^{K} \sigma(g(X^n) \cdot f(S_t) - \mathbb{I}_{X_i}[X^n]) \cdot g(X^n), \end{aligned} \quad (4)$$

where $\mathbb{I}_{X_i}[X^n]$ is an indicator function to indicate whether X^n is a sequential pattern $X_i \in \mathcal{F}(S_t)$ (i.e., the negative sequential pattern appears in the target sequence S_t) and when $n = 0$, then $X^n = X_i$.

3.3 Sqn2Vec Method for Learning Sequence Embeddings

When associating a sequence S_t with a set of SPs, S_t may not contain any SPs. In this case, we cannot learn a meaningful embedding vector for S_t. To avoid this problem, we propose two models which combine information of both single symbols and SPs to learn embedding vectors for sequences. These two models named **Sqn2Vec-SEP** and **Sqn2Vec-SIM** are presented next.

Sqn2Vec-SEP Model to Learn Sequence Embeddings. Given a sequence S_t, we separately learn an embedding vector $f_1(S_t)$ for S_t based on its symbols using the document embedding model PV-DBOW [10] and an embedding vector $f_2(S_t)$ for S_t based on its SPs (see Sect. 3.2). We then take the average of two embedding vectors to obtain the final embedding vector $f(S_t) = \frac{f_1(S_t)+f_2(S_t)}{2}$ for that sequence. The basic idea of **Sqn2Vec-SEP** is illustrated in Fig. 2.

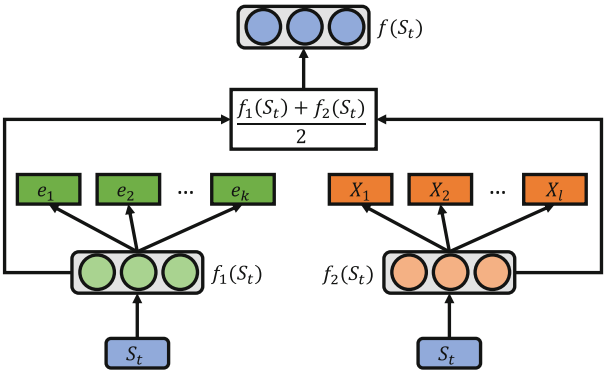


Fig. 2. Sqn2Vec-SEP model. Given a target sequence S_t, we learn the embedding vector $f_1(S_t)$ to predict its belonging symbols and learn the embedding vector $f_2(S_t)$ to predict its belonging SPs. We then take the average of $f_1(S_t)$ and $f_2(S_t)$ to obtain the final embedding vector $f(S_t)$ for S_t.

Sqn2Vec-SIM Model to Learn Sequence Embeddings. In the **Sqn2Vec-SEP** model, the sequence embeddings only capture the latent relationships between sequences and symbols and those between sequences and SPs separately. To overcome this weakness, we further propose the **Sqn2Vec-SIM** model which uses information of both single symbols and SPs of a sequence simultaneously. The overview of this model is shown in Fig. 3. More specifically, given a sequence S_t, our goal is to minimize the following objective function:

$$\mathcal{O}_2 = -\left[\sum_{e_i \in \mathcal{I}(S_t)} \log \Pr(e_i \mid S_t) + \sum_{X_i \in \mathcal{F}(S_t)} \log \Pr(X_i \mid S_t)\right], \tag{5}$$

where $\mathcal{I}(S_t)$ is the set of singleton symbols contained in S_t and $\mathcal{F}(S_t)$ is the set of SPs contained in S_t.

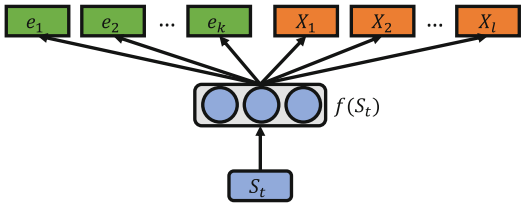


Fig. 3. Sqn2Vec-SIM model. Given a target sequence S_t, $\mathcal{I}(S_t) = \{e_1, e_2, ..., e_k\}$ is the set of symbols contained in S_t and $\mathcal{F}(S_t) = \{X_1, X_2, ..., X_l\}$ is the set of SPs contained in S_t. We learn the embedding vector $f(S_t)$ for S_t to predict both its belonging symbols and SPs.

Equation 5 can be simplified to:

$$\mathcal{O}_2 = - \sum_{p_i \in \mathcal{I}(S_t) \cup \mathcal{F}(S_t)} \log \Pr(p_i \mid S_t), \tag{6}$$

where $p_i \subseteq S_t$ is a symbol or a sequential pattern.

Following the same procedure in Sect. 3.2, we learn the embedding vector $f(S_t)$ for S_t, and the embedding vectors of two sequences S_i and S_j are close to each other if they contain similar symbols and SPs.

4 Experiments

4.1 Sequence Classification

The first experiment focuses on sequence classification, in which we compare our method with 11 baselines using sequential data from four application domains: text mining, action recognition, navigation analysis, and system diagnosis.

Datasets. We use eight benchmark datasets which are widely used for sequence classification. Their characteristics are summarized in Table 1. The *reuters* dataset is the four largest subsets of the Reuters-21578 dataset, consisting of news stories [19]. The three datasets *aslbu*, *aslgt*, and *auslan2* are derived from the videos of American and Australian Sign Language expressions [6]. The *context* dataset presents different locations of mobile devices carried by end-users [12]. The two datasets *pioneer* and *skating* are used in action recognition, which were introduced in [6]. The final dataset *unix* contains the command-line histories in a Unix system of nine end-users [19]. All datasets were used to evaluate the accuracy of sequence classification in [4–6,9,19].

Table 1. Statistics of eight sequential datasets.

Dataset	# sequences	# symbols	min. len	max. len	avg. len	# classes
reuters	1,010	6,380	4	533	93.84	4
aslbu	424	250	2	54	13.05	7
aslgt	3,464	94	2	176	43.67	40
auslan2	200	16	2	18	5.53	10
context	240	94	22	246	88.39	5
pioneer	160	178	4	100	40.14	3
skating	530	82	18	240	48.12	7
unix	5,472	1,697	1	1,400	32.34	4

Baselines. For a comprehensive comparison, we employ 11 state-of-the-art up-to-date baselines which can be categorized into four main groups:

- **Unsupervised SP-based methods:** We compare our method – **Sqn2Vec** with two state-of-the-art methods GoKrimp [9] and ISM [5]. We adopt their classification performances from their corresponding papers. We also employ another unsupervised baseline which constructs a binary feature vector for each sequence, with components indicating whether this sequence contains a sequential pattern with a $\triangle$-gap constraint (see Sect. 3.2). We name this baseline SP-BIN.
- **Supervised SP-based methods:** We select three representative and up-to-date baselines, namely BIDE-DC [6], SCIP [19], and MiSeRe [4]. We adopt the classification results of BIDE-DC reported in its supplemental appendix[1], those of SCIP from Table 10 in [19] and Fig. 12 in [4], and those of MiSeRe from Fig. 8 in [4].
- **Unsupervised embedding methods:** By considering a sequence as a document and symbols as words, we apply two recent state-of-the-art document embedding models for learning sequence embeddings, which are PV-DBOW [10] and Doc2Vec-C [2]. We also learn embedding vectors for sequences based on SPs (see Sect. 3.2), which we name Doc2Vec-SP.
- **Supervised embedding methods:** We implement two deep recurrent neural network models for sequence classification, LSTM and Bi-LSTM [16].

Our method **Sqn2Vec** has two different models which use different combinations of symbols and SPs. The **Sqn2Vec-SEP** model learns sequence embedding vectors from symbols and SPs separately (see Sect. 3.3) while the **Sqn2Vec-SIM** model learns sequence embedding vectors from symbols and SPs simultaneously (see Sect. 3.3).

Evaluation Metrics. After the feature vectors of sequences are constructed or learned, we feed them to an SVM with linear kernel [1] to classify the sequence labels. We use the linear-kernel SVM (a simple classifier) since we focus on the sequence embedding learning, not on a classifier, and this classifier was also used in [4,5,9,19]. The hyper-parameter C of SVM is set to 1, the same setting used in previous studies [5,9,19]. Each dataset is randomly split into 9 folds for training and 1 fold for testing. We repeat the classification process on each dataset 10 times and report the average classification accuracy. The standard deviation is not reported since all methods are very stable (their standard deviations are less than 10^{-1}).

Parameter Settings. Our method **Sqn2Vec** has three important parameters: the minimum support threshold δ, the gap constraint $\triangle$ for discovering SPs and the embedding dimension d for learning sequence embeddings. Since we develop **Sqn2Vec** in a fully unsupervised learning fashion, the values for δ, $\triangle$, and d

[1] https://sites.google.com/site/dfradkin/kais2014-separateAppendix.pdf.

are assigned without using sequence labels. We set $d = 128$ (a common value used in embedding methods [7,14]), set $\triangle = 4$ (a small gap which is able to capture the context of each symbol [15]), and set δ following the *elbow method* in [15]. Figure 4 illustrates the elbow method. From the figure, we can see that when the δ value decreases, the number of SPs slightly increases until a δ value where it significantly increases. This δ value, highlighted in red in the figure and chosen by the elbow method without considering the labels of sequences, is used in our experiments. In Sect. 4.1, we analyze the potential impact of selecting three parameters δ, $\triangle$, and d on the classification performance.

For a fair comparison, we use the same minimum support thresholds and gap constraints for our method and the baseline SP-BIN. We also set the embedding dimension required by three baselines PV-DBOW, Doc2Vec-C, and Doc2Vec-SP to 128, the same as one used in our method. For Doc2Vec-C, we use the source code[2] provided by the author with the same parameter settings except $d = 128$. We implement LSTM and Bi-LSTM with the following details[3]: the dimension of symbol embedding is 128, the number of LSTM hidden units is 100, the number of epochs is 50, the mini batch size is 64, the dropout rate for symbol embedding and LSTM is 0.2, and the optimizer is Adam.

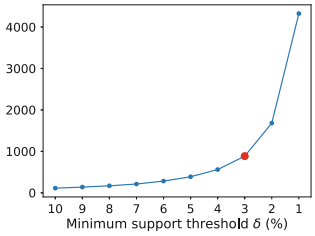


Fig. 4. The number of SPs discovered from the *reuters* dataset per δ (here, $\triangle = 4$). The red dot indicates the δ value selected via the elbow method. (Color figure online)

Results and Discussion. From Table 2, we can see two models in our method **Sqn2Vec** clearly results in better classification on all datasets compared with unsupervised embedding methods. **Sqn2Vec-SEP** achieves 2–97%, 5–232%, and 1–17% improvements over PV-DBOW, Doc2Vec-C, and Doc2Vec-SP respectively. As discussed in Sect. 1, two document embedding methods PV-DBOW and Doc2Vec-C perform poorly on sequential datasets with the small number of symbols, namely *aslbu*, *aslgt*, *auslan2*, *context*, *pioneer*, and *skating*. Especially, on the dataset *auslan2* whose the vocabulary size is only 16, their performances dramatically reduce, where they are 97–232% and 106–247% worse than **Sqn2Vec-SEP** and **Sqn2Vec-SIM** respectively. In contrast, on *unix* and the text dataset *reuters*, where the vocabulary size is large enough, their performances are quite good. Doc2Vec-C even achieves the second best result on *reuters*.

On all the datasets with the small vocabulary size, Doc2Vec-SP significantly outperforms PV-DBOW and Doc2Vec-C. This demonstrates that learning sequence embeddings from SPs is more effective than learning sequence embeddings from symbols, as discussed in Sect. 3.2. Two our models (**Sqn2Vec-SEP** and **Sqn2Vec-SIM**) are always superior than Doc2Vec-SP. This proves that our

[2] https://github.com/mchen24/iclr2017.

[3] We use the parameter settings suggested by Keras (https://keras.io) for LSTM.

proposal to incorporate the information of both singleton symbols and SPs into the sequence embedding learning is a better strategy than learning the sequence embeddings from SPs only (see Sect. 3.3).

For most cases, our method **Sqn2Vec** is better than unsupervised pattern-based methods. On three datasets *auslan2*, *skating*, and *unix*, **Sqn2Vec-SIM** outperforms these approaches by large margins (achieving 7–17%, 12–32%, and 33–34% gains over SP-BIN, GoKrimp, and ISM). Interestingly, our developed baseline SP-BIN, which uses SPs with a $\triangle$-gap constraint, is generally better than two state-of-the-art methods GoKrimp and ISM. This verifies our intuition in Sect. 3.2 that SPs satisfying a $\triangle$-gap constraint are more meaningful and discriminative since they can capture the context of each symbol.

Compared with supervised pattern-based methods and supervised embedding methods, our **Sqn2Vec** produces comparable performances on most datasets. It outperforms BIDE-DC, SCIP, and deep recurrent neural networks (LSTM and Bi-LSTM) on all datasets except *aslgt* and *unix*. Note that these methods leverage the labels of sequences when they construct/learn sequence representations, an impractical condition which actually benefits the supervised methods.

Table 2. Accuracy of our **Sqn2Vec** and 11 baselines on eight sequential datasets. Bold font marks the best performance in a column. The last row denotes the δ values used by our method for each dataset; they are determined using the elbow method (see Fig. 4). "–" means the accuracy is not available in the original paper.

Method	*reuters*	*aslbu*	*aslgt*	*auslan2*	*context*	*pioneer*	*skating*	*unix*
SP-BIN	94.85	**63.49**	78.76	28.00	90.83	**100.00**	31.70	82.06
GoKrimp	–	59.86	81.90	29.50	82.08	**100.00**	25.80[a]	74.33
ISM	–	60.20	75.70	24.60	78.30	**100.00**	25.60	–
BIDE-DC	–	59.03	**82.30**	30.67	51.53	97.92	28.11	57.74
SCIP	96.63	59.00	65.00	32.00	00.00	97.00	26.00	88.96
MiSeRe	–	63.00	74.00	32.00	81.00	**100.00**	31.00	–
PV-DBOW	95.84	46.51	70.98	16.00	86.25	84.38	26.79	90.75
Doc2Vec-C	97.62	37.21	49.25	9.50	37.50	62.50	16.98	86.35
Doc2Vec-SP	97.13	60.93	78.67	30.50	87.08	98.75	31.32	83.18
LSTM	90.40	50.70	69.74	25.00	71.25	93.75	27.74	90.86
Bi-LSTM	91.39	56.74	72.74	29.50	78.75	94.38	32.83	**91.79**
Sqn2Vec-SEP	**98.02**	62.56	81.18	31.50	**91.67**	99.38	**36.60**	90.78
Sqn2Vec-SIM	94.95	62.09	78.90	**33.00**	87.50	**100.00**	33.96	89.40
δ (%)	3%	2%	5%	3%	20%	4%	10%	7%

[a] As GoKrimp uses another version of *skating*, its result for *skating* is copied from [5].

Parameter Sensitivity. We examine how the different choices of three parameters δ, $\triangle$, and d affect the classification performance of **Sqn2Vec-SEP** on five datasets *reuters*, *aslbu*, *aslgt*, *auslan2*, and *pioneer*. Figure 5 shows the classification results as a function of one chosen parameter when the others are set to their default values. From Fig. 5(a), we can see our method is very stable on

two datasets *reuters* and *aslgt*, where its classification performance just slightly changes with different δ values. On three datasets *aslbu*, *auslan2*, and *pioneer*, our prediction performance shows an increasing trend as δ is decreased. Another observation is that the values for δ selected by the elbow method often lead to the best or close to the best accuracy.

From Fig. 5(b), we also observe the performance of our **Sqn2Vec-SEP** is consistent on *reuters* and *aslgt*, where the gap constraint $\triangle$ is gain of relatively little relevant to the predictive task. On contrary, there is a first-increasing and then-decreasing accuracy line on two datasets *aslbu* and *pioneer*. One possible explanation is that if we set $\triangle$ large, the generated SPs are less meaningful as we discussed in Sect. 3.2. On *auslan2*, the increase of accuracy converges when $\triangle$ reaches 4.

Figure 5(c) suggests that the predictive performance increases on two datasets *aslgt* and *auslan2* when d is increased whereas there is a first-increasing and then-decreasing accuracy line on *aslbu* and *pioneer*. This finding differs from those in document embedding methods, where the embedding dimension generally shows a positive effect on document classification [2]. Again, our predictive performance is steady on *reuters*, which is shown by a straight accuracy line.

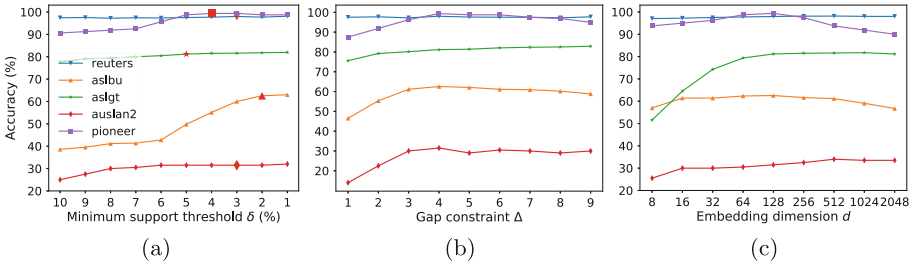


Fig. 5. Parameter sensitivity in sequence classification on five datasets *reuters*, *aslbu*, *aslgt*, *auslan2*, and *pioneer*. The minimum support thresholds δ selected via the elbow method and used in our experiments are indicated by red markers. (Color figure online)

4.2 Sequence Clustering

The second experiment illustrates how the latent representations learned by our proposed method can help the sequence clustering task, wherein we compare its performance with those of four baselines using text data.

Datasets. We use two text datasets for sequence clustering, namely *webkb* [15] and *news* [19]. *webkb* contains the content of webpages collected from computer departments of various universities. *news* is a subset of the dataset 20newsgroup, which is generated by selecting the five largest groups of documents. These two datasets are normalized (i.e., stop words are removed and the remaining words are stemmed), and can be downloaded from this website[4]. Their properties are summarized in Table 3.

[4] http://ana.cachopo.org/datasets-for-single-label-text-categorization.

Table 3. Statistics of two text datasets.

Dataset	# sequences	# symbols	min. len	max. len	avg. len	# classes
webkb	4,168	7,770	1	20,628	134.35	4
news	4,976	27,881	1	6,779	140.07	5

Baselines. We compare our **Sqn2Vec** with state-of-the-art embedding methods in text (PV-DBOW, Doc2Vec-C, and Doc2Vec-SP) and an unsupervised pattern-based method SP-BIN. We choose SP-BIN because it always outperforms other unsupervised pattern-based methods in sequence classification. These baselines are introduced in Sect. 4.1. We exclude supervised methods since they require sequence labels during the learning process, thus inappropriate for our unsupervised learning task – clustering.

Evaluation Metrics. To evaluate the clustering performance, the embedding vectors provided by each method are input to a clustering algorithm. Here, we use K-means (a simple clustering method) to group data and assess the clustering results in terms of mutual-information (MI) and normalized mutual-information (NMI). We conduct clustering experiments 10 times. We then report the average and standard deviation of clustering performance.

Parameter Settings. We use the same parameter settings as in sequence classification for four baselines and our method **Sqn2Vec**, except that the values for δ are selected using the elbow method (see Fig. 4).

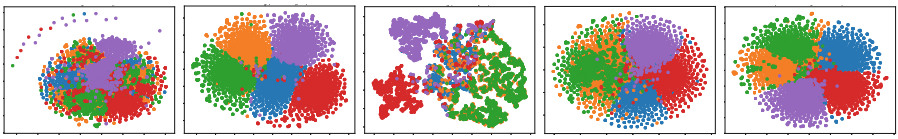
Results and Discussion. From Table 4, we can see **Sqn2Vec** outperforms all the competitive baselines in terms of both MI and NMI. Compared with state-of-the-art document embedding methods, **Sqn2Vec-SEP** outperforms PV-DBOW, Doc2Vec-C, and Doc2Vec-SP by 24–32%, 38–40%, and 5–8% when clustering the *webkb* dataset. Similar improvements can be observed when clustering the *news* dataset, where the gains obtained by **Sqn2Vec-SEP** over PV-DBOW, Doc2Vec-C, and Doc2Vec-SP, around 8–14%, 72–77%, and 34–40%. Compared with the pattern-based method, the improvements are more significant, where **Sqn2Vec-SEP** outperforms SP-BIN by 163–184% on *webkb* and 767–1,090% on *news*.

4.3 Sequence Visualization

Figure 6 visualizes the document reprsentations learned by SP-BIN, PV-DBOW, Doc2Vec-C, Doc2Vec-SP, and our **Sqn2Vec-SEP** on the *news* dataset. We can see the documents from the same categories are clearly clustered using the embeddings generated by PV-DBOW and **Sqn2Vec-SEP**. On the other hand, SP-BIN, Doc2Vec-C, and Doc2Vec-SP do not distinguish different categories clearly.

Table 4. MI and NMI scores of our method **Sqn2Vec** and four baselines on two text datasets. The MI score is a non-negative value while the NMI score lies in the range $[0, 1]$. Bold font marks the best performance in a column.

Dataset	*webkb*		*news*	
Method	MI	NMI	MI	NMI
SP-BIN	0.19 (0.06)	0.16 (0.05)	0.10 (0.04)	0.09 (0.03)
PV-DBOW	0.41 (0.12)	0.34 (0.10)	1.04 (0.11)	0.72 (0.06)
Doc2Vec-C	0.39 (0.02)	0.30 (0.01)	0.69 (0.05)	0.44 (0.03)
Doc2Vec-SP	0.50 (0.14)	0.40 (0.10)	0.85 (0.12)	0.58 (0.07)
Sqn2Vec-SEP	**0.54 (0.10)**	**0.42 (0.07)**	**1.19 (0.15)**	**0.78 (0.06)**
Sqn2Vec-SIM	0.51 (0.11)	0.41 (0.08)	1.08 (0.22)	0.71 (0.12)
δ (%)	3%		3%	



(a) SP-BIN (b) PV-DBOW (c) Doc2Vec-C (d) Doc2Vec-SP (e) Sqn2Vec-SEP

Fig. 6. Visualization of document embeddings on *news* using t-SNE [11]. Different colors represent different categories. (Color figure online)

5 Conclusion

We have introduced **Sqn2Vec** – an unsupervised method for learning sequence embeddings from information of both singleton symbols and SPs. Our method is capable of capturing both the sequential relation among symbols and the semantic similarity among sequences. Our comprehensive experiments on 10 standard sequential datasets demonstrated the meaningful and discriminative representations learned by our approach in both sequence classification and sequence clustering tasks. In particularly, **Sqn2Vec** significantly outperforms several state-of-the-art baselines including pattern-based methods, embedding methods, and deep neural network models. Our approach can be applied to different real-world applications such as text mining, bio-informatics, action recognition, and system diagnosis. One of our future works is to integrate SPs into deep neural network models, e.g., LSTM and Bi-LSTM, to improve the classification performance.

Acknowledgment. Dinh Phung and Tu Dinh Nguyen gratefully acknowledge the partial support from the Australian Research Council (ARC).

References

1. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**(3), 1–27 (2011)
2. Chen, M.: Efficient vector representation for documents through corruption. In: ICLR (2017)
3. De Smedt, J., Deeva, G., De Weerdt, J.: Behavioral constraint template-based sequence classification. In: Ceci, M., Hollmén, J., Todorovski, L., Vens, C., Džeroski, S. (eds.) ECML PKDD 2017. LNCS (LNAI), vol. 10535, pp. 20–36. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71246-8_2
4. Egho, E., Gay, D., Boullé, M., Voisine, N., Clérot, F.: A user parameter-free approach for mining robust sequential classification rules. Knowl. Inf. Syst. **52**(1), 53–81 (2017)
5. Fowkes J., Sutton, C.: A subsequence interleaving model for sequential pattern mining. In: KDD, pp. 835–844 (2016)
6. Fradkin, D., Mörchen, F.: Mining sequential patterns for classification. Knowl. Inf. Syst. **45**(3), 731–749 (2015)
7. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: KDD, pp. 855–864 (2016)
8. Jin, L., Schuler, W.: A comparison of word similarity performance using explanatory and non-explanatory texts. In: NACACL, pp. 990–994 (2015)
9. Lam, H.T., Mörchen, F., Fradkin, D., Calders, T.: Mining compressing sequential patterns. Stat. Anal. Data Mining ASA Data Sci. J. **7**(1), 34–52 (2014)
10. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: ICML, pp. 1188–1196 (2014)
11. Van Der Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**, 2579–2605 (2008)
12. Mäntyjärvi, J., Himberg, J., Kangas, P., Tuomela, U., Huuskonen, P.: Sensor signal data set for exploring context recognition of mobile devices. In: PerCom, pp. 18–23 (2004)
13. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: NIPS, pp. 3111–3119 (2013)
14. Nguyen, D., Luo, W., Nguyen, T.D., Venkatesh, S., Phung, D.: Learning graph representation via frequent subgraphs. In: SDM, pp. 306–314 (2018)
15. Rousseau, F., Kiagias, E., Vazirgiannis, M.: Text categorization as a graph classification problem. In: ACL, pp. 1702–1712 (2015)
16. Tai, K.S., Socher, R., Manning, C.: Improved semantic representations from tree-structured long short-term memory networks. In: ACL, pp. 1556–1566 (2015)
17. Wang, J., Han, J.: BIDE: efficient mining of frequent closed sequences. In: ICDE, pp. 79–90 (2004)
18. Zaki, M., Meira, W.: Data Mining and Analysis: Fundamental Concepts and Algorithms. Cambridge University Press, Cambridge (2014)
19. Zhou, C., Cule, B., Goethals, B.: Pattern based sequence classification. IEEE Trans. Knowl. Data Eng. **28**(5), 1285–1298 (2016)