Construction of Efficient V-Gram Dictionary for Sequential Data Analysis

Igor Kuralenok
JetBrains Research
ikuralenok@gmail.com

Natalia Starikova
First Line Software
natash.starikova@gmail.com

Aleksandr Khvorov
St.Petersburg State University
khvorov.aleksandr@gmail.com

Julian Serdyuk
St.Petersburg State University
masyes@mail.com

ABSTRACT

This paper presents a new method for constructing an optimal feature set from sequential data. It creates a dictionary of n-grams of variable length (we call them v-grams), based on the minimum description length principle. The proposed method is a dictionary coder and works simultaneously as both a compression algorithm and as unsupervised feature extraction. The length of constructed v-grams is not limited by any bound and exceeds 100 characters in provided experiments. Constructed v-grams can be used for any sequential data analysis and allows transfer bag-of-word techniques to non-text data types. The method demonstrates a high compression rate on various real-life datasets. Extracted features generate a practical basis for text classification, that shows competitive results on standard text classification collections without using the text structure. Combining extracted character v-grams with the words from the original text we achieved substantially better classification quality than on words or v-grams alone.

KEYWORDS

Dictionary coder, v-gram, minimum description length principle, character-based n-gram, text compression, unsupervised feature extraction, text classification

ACM Reference Format:
Igor Kuralenok, Natalia Starikova, Aleksandr Khvorov, and Julian Serdyuk. 2018. Construction of Efficient V-Gram Dictionary for Sequential Data Analysis. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271789

1 INTRODUCTION

The importance of sequential data analysis nowadays is hard to overestimate. This type of tasks comes from many fields: text retrieval, speech recognition, genetic data analysis, antifraud, etc.. Historically the most competitive field for such tasks was information retrieval. Once build, methods from this area of computer science then often migrate to other fields. One of the limiting factors of such migration is that many IR techniques use word representation as their basis. Once the method extracts a set of descriptive subsequences they could be used as words. This approach is widely used in biology, clickstream analysis, etc.. However the method for extracting subsequences in most of the cases is straightforward n-gram analysis. This simplification of the extraction method has its downsides: the curse of dimensionality, performance difficulties with substantial dictionary sizes, etc. Feature extraction/construction methods could help to overcome these problems.

There are two aspects of the quality of the feature extraction method in this context: the quality of the extracted feature set and the ability of the method to describe the data. We can evaluate the first aspect on some known task such as text classification, using no information on text structure (e.g., removing all spaces from the source text). Other data types have no such information and methods should be able to work without it. Extracted subsequences from the text are called character n-grams. This type of data has proven its value [29], so despite the challenging task setup, we can expect competitive results.

Text classification is one of the most popular tasks in text retrieval [13]. Like any classification task in machine learning, it relies critically on a feature set quality [10]. In this paper, we fix the learning method (SVM) and then compare the quality of the constructed feature sets by classification quality of SVM based on them.

The second aspect we associate with the compression quality. If the feature set with entropy coding can compress the data effectively, then it contains independent blocks of data, which could be treated as "words". This binary structure of the task gives us two potential sources of ideas: one from compression field, the second from text retrieval. In Sec. 2 we will review the most interesting of them.

In this paper, we present a new method, based on the MDL principle. It can deliver competitive text classification quality and provide good compression rate on various data. Constructed limited optimal feature set is referred to as a dictionary and consists of character-based n-grams of a variable length. The term *'v-grams'* is introduced, emphasizing the variable and unlimited length of such n-grams.

The idea of the proposed method is similar to described in [21], but the instruments and the approach differ. Authors in [21] formulate document compression as a binary optimization task and use algebraic instruments to solve this problem, while the information theory approach applied to random processes is used in this paper. Besides the instruments, the difference comes from the nature of the optimization: Paskov et al. filter out uninformative n-grams (like backward elimination), while in this paper we construct informative (like forward selection). Our approach does not limit the problem space allowing us to construct n-grams of potentially unlimited length.

The main contributions of this paper:

- Approach to sequence generation modeling that theoretically converges to optimal dictionary based representation.
- Implementing our approach in a real-world environment.
- Asymmetric measure of terms similarity, based on the informativeness.
- Evaluation of the compression rate of the method on various data to demonstrate its ability to find proper terms for data description.
- Competitive results on text classification task without using information on text structure (no spaces, punctuation, etc.).

The rest of the paper is organized as follows. At first, we perform a quick review of existing approaches to feature extraction from sequential data. Then comes the description of our approach in Sec. 3. It consists of two main parts: the theoretical approach to the sequence modeling, and implementation of this approach on practice. Sec. 4 contains three types of experiments: the testing ability of the proposed method to restore fixed random process of sequences generation, testing compression quality on various data types and finally evaluation of the classification quality on textual data. Discussion on possible future work could be found in Sec. 5

2 RELATED WORK

2.1 Compression and machine learning

A connection between machine learning and data compression has been thoroughly studied over the last few years. Such an approach is based on the MDL principle [22]. In [8] authors present the application of the compression-based algorithm to different machine learning tasks, such as text categorization and similarity measures. Compression-based algorithms can also be applied to other tasks, such as hierarchical clustering [6], anomaly detection [14], spam filtering [4], language clustering and others [2].

In this paper, the primary focus is on the use of compression algorithms to feature extraction. In [23] authors present a comprehensive survey of compression-based similarity measures and present theoretical and empirical connections between traditional machine learning vector models and compression. The feature set, corresponding to the famous Lempel-Ziv algorithm (LZ77) [30] is described in this paper among others.

The LZ77 algorithm is the universal lossless compression algorithm based on the repeated occurrence of data. One of the disadvantages of this method is its low efficiency on small datasets. The idea *"to store substring found by Lempel-Ziv schemes as explicit features"* is provided in [23] as a future work suggestion. With using such substrings as features, the LZ77 algorithm acquires portability and would be able to work more efficient on small datasets. In this paper, we use a dictionary from a dictionary coder algorithm as features.

A similar idea of compression feature learning based on the MDL principle is presented in [21]. The intention of both methods is using the compression to feature extraction. While in [21] authors formulate the problem as the binary optimization, in this paper the same idea is observed from the information theory perspective. This idea allows taking into account interdependence between features.

Information theory and compression have a natural and robust bond. Intrinsically, the whole theoretical background for data compression is based on the information theory. As far as the paper is concerned with feature extraction, a short survey of information theory application for this task is provided.

2.2 Feature extraction with information theory

Information theory is widely used to feature extraction purposes. Applying such measures as mutual information, information gain, χ^2 and others for filtering methods is very popular, especially in text classification tasks [13, 20, 27]. Moreover, information-theoretical methods allow inspecting the interdependence between features, which is very important for both the size of the feature set and the accuracy of machine learning algorithms.

The application of information theory to feature extraction is useful when the question of feature redundancy arises. The information-theoretical measures are convenient to study features interdependence. The groundbreaking paper [5] provides a unifying framework to feature selection based on information theory, connecting the concepts of feature relevance and redundancy.

Another interesting approach to feature selection based on information theory is presented in [9]. Authors demonstrate that previous methods use unrealistic approximations. Authors introduced a new method based on a variational lower bound on mutual information and provided a new class of assumptions.

The most popular measure applied to feature extraction is mutual information. A comprehensive review of feature selection methods is provided in [28]. A new approach, presented in [5], associates mutual information criteria with a conditional likelihood problem, while authors in [9] provide a framework based on the lower bounds for mutual information. In most of the papers, the mutual information is used for calculating information between features and labels, while it can also be applied for measuring information between features.

There are three main differences between the proposed method from the other information-theoretic methods mentioned above. First, the proposed method is unsupervised feature extraction, thus can be applied for both supervised and unsupervised algorithms. Next, mutual information, which is utilized in the majority of other papers, is symmetrical measure, while in this paper cross-entropy is used, removing constraints on the symmetry. Finally, another critical difference is that method constructs an optimal *set of features* for a processed dataset, unlike just selection of most correlated with the target (or relevant for information retrieval tasks) features. This difference allows diminishing the redundancy of the information presented in the features.

The approach for constructing an optimal *set of features* is essential. As contrasted with other methods it allows to control the size of the optimal dictionary directly, as opposed to some stopping criterion [11]. Such stopping criteria might demand unreasonably big size of the dictionary, thus enlarging the size of feature set and the overall performance. Proposed approach pushes the boundaries in feature extraction algorithms from a practical perspective.

In the experimental section, we will show, that the proposed method can be successfully applied to text classification tasks. Feature selection based on the information theory measures occurs widely in text retrieval tasks. Authors in [7] provide a study on compression-based dissimilarity measures for text classification. An interesting approach is presented in [26], where authors examine Kullback-Leibler divergence and Jeffreys divergence in application to text categorization and introduce a new divergence measure, explicitly constructed for the multi-classification task.

One of the approaches to text classification includes utilizing n-grams as features. In [23] authors provide a connection of compression algorithms with n-grams. In this paper, character-based n-grams naturally occur as features.

2.3 Character-based n-grams

The utilization of the n-gram model for text classification is quite widespread, word-based n-grams serve as a well-established tool. Recently a bunch of papers on character-based n-grams appeared. Perhaps one of the most popular papers on this topic is [29], which describes the use of character n-grams in addition to word-based ones. This approach is applied to rare words in neural machine translation.

A combination of word-based n-grams with character-based occurs in other papers. In [3] character n-grams are utilized as subword information. Subword units are also used to obtain representations of rare words in [24]. In [25], a language model based on restricted Boltzmann machines, where words are encoded as a set of character-based n-grams, is introduced.

The application of n-grams as features enlarges the size of feature set tremendously, thus requiring feature extraction methods. Proposed in [21] compression-based method works as a feature selection, being applied to all n-grams with the length less than 6. In most papers, the length of n-grams is limited to some fixed number, even when it is stated to be variable. It seems reasonable for word-based n-grams to have a short length. However, limiting the length of character-based n-grams is a different matter. Although n-grams proposed in [3] are character-based, their length is limited from 3 to 6.

The main difference between the proposed method from other methods utilizing character-based n-grams in parametrization. The proposed method can build a dictionary of any fixed size, and not limited by the length of n-grams. This setting is particularly important and provides the construction of useful features independent on the length.

3 PROPOSED METHOD

3.1 Background

Let $\mathcal{A}$ be an arbitrary alphabet. Denote as $X = \{x\}$ a set of entries x based on alphabet $\mathcal{A}$:

$$x = \{\chi_k\}_{k=1}^{|x|}, \chi_k \in \mathcal{A}, x \in X$$

Assume that entries $x \in X$ were generated by some fixed random process P. Dictionary A consists of terms $a = \{\alpha_j\}_{j=1}^{|a|}, \alpha_j \in \mathcal{A}$

Table 1: Notations table

Notation	Description
$\mathcal{A}$	alphabet of allowed symbols
x	sequence from the training set
X	training set of sequences
$\|x\|$	power of set or symbol length of the sequence
$\|x\|_A$	length of the x sequence in terms of current dictionary |A|
$\|X\|_A = \sum_{x \in X} \|x\|_A$	total length of data set representaton in dictionary A
P	random process generating X
A	dictionary of subsequences of X
a, b	elements of the dictionary
$\{x_k\}_{t=0}^{\|x\|_A} = x$	items from the dictionary A, coding sequence x from the training set
$p(x_k \| A, P)$	probability of t-th coding component for fixed dictionary and P
$p(b \| a, A, P)$	probability of term b after term a occured
$f(a, b \| A, X_t)$	frequency of pair (a, b) in t-th re-sampling of X

Following the information theory, the process P can be modeled in terms of dictionary A if it contains the alphabet ($A \supset \mathcal{A}$) without loss of information[1].

$$x = \{x_k\}_{k=1}^{|x|_A}, x_k \in A$$

If for each term $a \in A$ we know the probability $p(a|A, P)$ of its generation by process P, then the total code length of X in terms of A will be the cross-entropy:

$$H(X, A) = \sum_{x \in X} \sum_{k=1}^{|x|_A} p(x_k | x_{k-1}, \dots, x_1, A, P) \log p(x_k | A, P) \qquad (1)$$

where $p(x_{ik} | x_{ik-1}, \dots, x_{i1}, A, P)$ is a true probability of generating x_{ik}, preceeded by terms $x_{i1}, \dots x_{ik-1}$, by process P. Note, that in $p(a|A, P)$ we can not ommit A, because the probability of the term depends on other terms in the dictionary. Assuming Markov property of P in terms of A, (1) can be reformulated in the following way:

$$H_M(X, A) = \sum_{x \in X} \sum_{k=1}^{|x|_A} p(x_k | x_{k-1}, A, P) \log p(x_k | A, P) \qquad (2)$$

The better A describes P, the more P becomes markovian: in terms of perfect A all terms are independent.

On the other hand, cross-entropy by definition equals to:

$$H_M(X, A) = H_M(X) + D_{KL}\left(p(b|a, A, P) || p(b|A, P)\right) \qquad (3)$$

where D_{KL} is Kullback-Liebler divergence. In this problem setting Kullback-Liebler divergence reflects the information that is lost in the process of coding, so-called *informativeness* further in the paper. The entropy term $H_M(X)$ does not depend on dictionary and could be excluded from the optimization. The most informative

[1]There could be more then one coding of the same x. If $x = abb$, $A = \{a, b, ab\}$, then there are two such variants: $\{a, b, b\}$ and $\{ab, b\}$. We use the one with minimal code length.

```
Data: Set of sequences $X$
Result: Optimal $A$
$A_0 = \mathcal{A}$;
$X_0$ = frequencies of alphabet terms in $X$;
$t = 0$;
repeat
    $t = t + 1$;
    $X_t = \emptyset$;
    repeat
        Pick $x$ at random from $X$;
        Split $x \longrightarrow \{x_k\}$ using terms from $A_{t-1}$ and
          frequencies from $X_{t-1}$;
        foreach unigram $a$ and bigram $(a, b)$ from $\{x_k\}$ do
          put $a \longrightarrow X_t$ and $(a, b) \longrightarrow X_t$ ;
    until $StoppingCriterion(X_t)$;
    if $t \bmod 2 = 1$ then
        $A_t = Expand(A_{t-1}, X_t)$;
    else
        $A_t = Reduce(A_{t-1}, X_t)$;
    end
until $t \bmod 2 = 0$ and $A_t \cap A_{t+1} > N$;
```

Algorithm 1: General schema of the algorithm

dictionary $\hat{A}$ describing random process P can be found by the following optimization problem:

$$\begin{aligned}\hat{A} &= \arg\min_A H(X, A) = \arg\min_A D_{KL}\left(p(b|a, A, P)||p(b|A, P)\right)\\ &= \arg\min_A \sum_{x\in X}\sum_{k=1}^{|x|_A} p(x_k|x_{k-1}, A, P)\log\frac{p(x_k|x_{k-1}, A, P)}{p(x_k|A, P)}\\ &= \arg\min_A \sum_{x\in X}\sum_{k=1}^{|x|_A} p(x_k|x_{k-1}, A, P)\log\frac{p(x_k, x_{k-1}|A, P)}{p(x_k|A, P)p(x_{k-1}|A, P)}\\ &= \arg\min_A \sum_{(a,b)\in A^2} f(a, b|A, X)p(b|a, A, P)\log\frac{p(a, b|A, P)}{p(a|A, P)p(b|A, P)}\end{aligned} \quad (4)$$

In the last transition we have grouped all entrances of pair (a, b) in sequences of X. $f(a, b|A, X)$—is the total frequency of (a, b) in the training set. We will minimize this formula by iteretively changing A. On each step we are greedily eliminating its biggest term by apending (a, b) into the dictionary. If the size of A is above wanted volume, reduce it back removing least informative term. This algorithm has nice convergence properties if $p(*|A, P)$ are precise: cross-entropy in our form is submodular [1], and the algorithm is greedy on the code length. The problem comes from the fact that these probabilities are not precise in practice, and we need to do some tricks to optimize the alphabet on real data.

3.2 Algorithm of v-grams extraction

The general schema of dictionary A construction (Alg. 1) consists of two phases, repeating iteratively:

(1) **Expansion phase.** Populating dictionary with the most informative elements.
(2) **Reduction phase.** Deleting the least informative elements from the dictionary.

Dictionary is examined on each iteration. Let $A_0 = \mathcal{A}$ denote the initial dictionary in the beginning of process, and A_t be the dictionary on t-th iteration. Let A'_t denote the expanded version of A_t. On each iteration t dictionary A_t is expanded to A'_t at the expansion phase and then is pruned to A_{t+1} at the reduction phase.

Equality $A_{t+1} = A_t$ reflects the stopping criteria of the whole process, meaning that dictionary remains unchanged after expansion and reduction phases. Both phases require statistical data on elements of A_t and A'_t. The expansion phase requires observed frequencies of all pairs of occurrences, while the reduction phase needs unigram frequencies. These statistics are calculated by sampling X with the common mechanism described below. New statistics on each phaser prevents overfitting on the observed sample set.

On each step, we deal with the dictionary A. Depending on a step and phase it can take values A_t or A'_t. To shorten notation, we will use term A instead of the full name of the dictionary. If this notation becomes unclear, we will use full naming.

Statistics calculation scheme: At this step we are given dictionary A, frequencies of the terms of the dictionary observed on previous statistics harvesting. With this data in hands we perform the following steps:

(1) Choose an element of the training set at random $x \in X$.
(2) Find the most compact representation of the element x in terms of the current dictionary A.
(3) Update frequencies of dictionary terms involved in this representation.
(4) Update frequencies for pairs of the dictionary elements that appear consecutively in the entry x.

Described steps are then repeated before stopping condition is reached. This part of the algorithm is the most computationally intensive, and we had to introduce the stopping condition to limit this computational difficulty. The idea behind the condition is that every element from A should be observed at least once with some fixed significance level.

For the set of drawn samples on the statistics calculation step, we use X_A notation if dictionary A were used for parsing.

Statistics calculation details. At the beginning of each statistics calculation step, we set all frequencies of elements of A to 0. With each drawn sequence x we increment frequencies of items, observed in the most compact representation of x.

As stated above, x always has at least one variant of representation in terms of A, but there could be more than one variant. To measure the quality of each variant, we can use clear metrics. The most simple is the number of elements in the representation of x. Despite the simplicity of this measure, it can be suboptimal regarding presentation code length optimized on later steps. We have experimented with this and several other metrics and found that the code length itself that requires additional data to be calculated is substantially better than the others. Code length requires probability values $p(a|A, P)$ on elements of A. We have used point estimates of these probabilities taken from the previous statistics calculation step. For the first step, all terms are assumed equally probable.

Another problem with the most compact representation is that it is computationally difficult to get for arbitrary A. The difficulty of the naive implementation is $O(2^{|x|-1})$. On the other hand, using information on A and frequency of its elements minimum code length could be found for $O(|x| log|A|)$. The algorithm has the following schema:

(1) Initialize array L of size $|x|$ by positive infinities
(2) For each position p from 0 to $|x| - 1$:
 (a) Find the longest entry a of A that is equal to substring of x starting from current position
 (b) For s from set of a and its "parents" update the value of $L[p + |s|]$ with the $L[p] + \log p(s|A,P)$ if it is less then current value

Resulting array L will contain the code length in the last cell and the proper presentation could be calculated from the backtrace.

Statistics calculation stopping criteria. During the process of statistics accumulation, a subset of textual data X denoted as X_t is examined. For representation of entries $x \in X_t$ elements from dictionary A are used. The goal is to calculate statistics for all elements of dictionary A in a reasonable time. We can formulate the condition on the statistics volume to observe the least probable dictionary element at least once with some fixed probability α.

The least probable dictionary element a_{min} can be found, using statistics gathered on previous step X_{t-1}:

$$p(a_i|A_t, X_{t-1}) = \frac{f(a_i|X_{t-1})}{|X_{t-1}|_{A_t}}$$

The minimal among probabilities from the previous step is selected. Corresponding to this probability element is denoted as a_{min}:

$$a_{min} = \arg\min_{a \in A_t} p(a|A_t, X_{t-1})$$

Let λ denote the following:

$$\lambda = p(a_{min}|A_t, X_{t-1}) \cdot |X_t|$$

Assume that elements of dictionary are appearing as a Poisson process, one can estimate the probability of element occurrence. With Poisson index $n = 0$, probability of event a_{min} never occurring is the following:

$$p(\{f(a_{min}) = 0\}|A_t, X_{t-1}) = e^{-\lambda} \cdot \frac{\lambda^n}{n!} = e^{-\lambda}$$

Setting up α as a confidence level to be small enough, the stopping criteria can be formulated as:

$$e^{-\lambda} = e^{p(a_{min}|A_t) \cdot |X_t|} < \alpha$$

Hence a condition on the current statistics volume $|X_t|$ appears:

$$|X_t| > -\frac{\log \alpha}{p(a_{min}|A_t)}$$

The size of re-sampling X_t in terms of the dictionary A_t serves as a stopping criteria of statistics accumulation.

Calculation of statistics precedes each of the main phases: expansion and reduction. After that, the random process P can be modeled by the data from X_t using calculated frequencies. Now the main phases of the algorithm can be explained in more details.

(1) **Expansion phase. Populating the dictionary with the most informative elements.** Most informative elements are estimated based on accumulated statistics for pairwise frequency and Kullback-Liebler divergence in the equation (4).

$$f(a,b|A,X_t)p(b|a,A,P) \log \frac{p(a,b|A,P)}{p(b|A,P)p(a|A,P)} \tag{5}$$

If we use point estimates of the probabilities of pair ab occurrence:

$$p(b|a,A,P) = \frac{f(a,b|X_t)}{|X_t|_A - |X_t|}$$

the error rate will be too high and we will always append pairs of infrequent terms. To prevent this we should do a bit more modeling.

Let x denote all elements of A except a and y denote all elements of A except b:

$$x = \{A_k\} \setminus a, y = \{A_k\} \setminus b$$

Then all possible cases of elements co-occurrence can be described as following pairs: ab, ay, xb, xy. These pairs can be examined as independent and incompatible random events that form a probabilistic simplex.

Let $f(a,b|A_k,X_k), f(a,y|A_k,X_k), f(x,b|A_k,X_k), f(x,y|A_k,X_k)$ be corresponding frequencies and $v_1 = p(a,b|A_k,P)$, $v_2 = p(a,y|A_k,P)$, $v_3 = p(x,b|A_k,P)$, $v_4 = p(x,y|A_k,P)$ denote corresponding probabilities. Denote resulting distribution as $V \sim \text{Dir}(\gamma_k)$, where components of $\gamma_{kuv} = f(u,v|A_k,X_k) + 1$.

Then v_1, v_2, v_3, v_4 can be used for calculating following probabilities from (5):

$$p(b|a,A_k,P) = \frac{v_1}{v_1 + v_2}$$

$$p(a,b|A_k,P) = v_1$$

$$p(b|A_k,P) = v_1 + v_3$$

$$p(a|A_k,P) = v_1 + v_2$$

Then expression under summation in (5) can be rewritten as

$$p(b|a,A_k,P) \log \frac{p(a,b|A_k,P)}{p(b|A_k,P)p(a|A_k,P)} = \frac{v_1}{v_1+v_2} \log \frac{v_1}{(v_1+v_3)(v_1+v_2)}$$

Finally, with applying integral smoothing, D_{KL} for pair (a,b) can be calculated in the following way:

$$\frac{D_{KL_{ab}}}{f(a,b|A_k,X_k)} = \int_V \frac{v_1}{v_1+v_2} \log \frac{v_1}{(v_1+v_3)(v_1+v_2)} d\text{Dir}(\gamma) \tag{6}$$

This integral is then calculated numerically drawing vectors v at random by $\text{Dir}(\gamma)$ distribution. After calculation of informativeness for all pairs using equation (6), these values are sorted. Elements with minimal D_{KL} are selected for expansion. Concatenated pairs are then used for dictionary expansion. Thus a length of dictionary terms can increase as 2^k with the number of iterations.

(2) **Reduction phase. Deleting the least informative elements from the dictionary.** For each dictionary element a_j except members of $\mathcal{A}$, we should calculate the cost of its removal. We code each element a by the rest of current dictionary $A_k \setminus a$, update statistics for coding elements $(a = (b_1, \ldots, b_s), b_j \in A_k \setminus a\ )$. During the update, we append to coding element frequencies the frequency of a. After the statistics update, the new code length is calculated. The described method gives us the upper bound on the X_k code length in terms of $A_k \setminus a$. Result values are then sorted and dictionary A is reduced to the required size N by excluding elements with minimal values.

Entries of constructed dictionary A are called v-grams with respect to variability of elements length. The algorithm scheme is similar to iterative conditional optimization. In this scheme a step in optimal direction is performed and its result is projected to the conditions. The both phases of the algorithm are greedy and optimize the same target function: the code length of X.

4 EXPERIMENTS

In this section results of experiments based on the proposed method are presented. Regarding the versatility of this method, it can be successfully used in a wide variety of applications. Since it is a feature extraction method based on the minimum description principle, it can be applied for compression and information retrieval tasks. These two are well-studied areas, and the effectiveness of the proposed approach will be demonstrated by encouraging results. It is worth noticing that such results are provided without any domain knowledge.

The rest of the section is organized as follows. In the first instance, it is demonstrated that assumptions from section 3 are consistent with the method's capability to restore original dictionary from dictionary-based sequences. Then we present a compression track to evaluate the effectiveness of constructed dictionary on various types of data. In the final section, the quality of text classification tasks on several public collections with v-grams used instead of words and in combination with them is studied.

4.1 Dictionary reconstruction

In the method description, a couple of assumptions were used. First, upper bound on code length instead of its true value to compare informativeness of entries is used. This assumption allows us to use the same calculated statistics for all terms of the dictionary, without it we will have to exclude each term one by one during exclude phase and calculate statistics once per term which is next to impossible in a computational sense. The second assumption simplifies calculation of KL-divergence D_{KL} with a new term from the dictionary (corresponding to the pair of old terms). It allows us to analyze each pair independently on each other; the alternative is to look for the combination of joins which is NP-hard. The following experiment on synthetic data is provided, to demonstrate that method is capable of reconstructing a fixed random dictionary so far. Experiment schema follows:

(1) Fix the alphabet $\mathcal{A}$ = Base64[2] encoding scheme.
(2) Set the dictionary A consisting of sequences of terms from the alphabet $\mathcal{A}$ according to the scheme below.
(3) Generate a set of strings: pick random entries from the dictionary uniformly and concatenate them.
(4) Direct the stream of generated strings to the input of the proposed method.
(5) Compare dictionary entries produced by the proposed method with dictionary A, checking exact match.

To set entries of A the following procedure is used:

(1) Choose a length of a dictionary entry using Poisson point process with $\lambda = 5$[3]. The length of a dictionary element $a \in A$ is equals to $|a| = \text{Poisson}(5) + 1$.
(2) Given a length of the dictionary entry $|a| = n$, construct an element itself, based on alphabet $\mathcal{A}$.
(3) Add obtained element to A.

Following described procedure, a dictionary A with length $|A| = 1\,000$ is obtained for experiments. Based on it, 10 000 sequences of length 1 000 as training data are generated. After that a dictionary of v-grams is constructed, trained on generated sequences. These results are compared to the original dictionary. A repetitive sequence of experiments shows that $99 \pm 1\%$ of dictionary A entries are present in the constructed dictionary. This fact demonstrates that introduced assumptions do not spoil theoretical base.

4.2 Sequential data compression

In this set of experiments, the proposed method is compared with other compression algorithms on various sequential data X. In this problem setting, the process of sequence generation is modeled, assuming that strings $x \in X$ are independent. This assumption characterizes streams of data, not batches of data. One of the approaches to batch compression includes preliminary sorting of data before the compression. This approach is beneficial in the majority of cases. The excellent compression rate can be reached, playing with data dependence.

Experiments are conducted to compare average compression rate of data X without additional preprocessing (like sorting). The original data X is shuffled in some random order, to achieve this goal. The proposed method is then combined with arithmetic coding. Constructed dictionary[4] is appended to data X, compressed by this dictionary. This pair acted as an experiment output and used to estimate the total size of the archive.

Genetic data compression. For this experiment the data from the Oxford Nanopore Human Reference Dataset [12] is used. Extracted FASTQ data from NA12878 genome[6] was compressed by different methods. Compression rates of the method, gzip and bzip2[7] are present at Tab. 2.

A persistent tendency of compression ratio and dictionary size correlation was found (Fig. 1). This fact provides evidence to one of the main characteristics of the method, namely constructing an optimal feature set. For a feature selection method, instead of

[2] https://tools.ietf.org/html/rfc4648

[3] This parameter value allows generating long (> 10 characters) entries in the dictionary A

[4] Compressed by gzip[5]

[6] https://github.com/nanopore-wgs-consortium/NA12878

[7] http://bzip.org/

Table 2: Genetic data compression

Method	Size, bytes	Compression rate, %
Original data	3 288 652 678	-
gzip	937 092 187	28.5%
bzip2	871 801 533	26.5%
Dictionary size 500	845 185 024	25.7%
Dictionary size 1 000	840 701 362	25.6%
Dictionary size 2 000	836 993 024	25.5%
Dictionary size 10 000	818 544 640	24.9%
Dictionary size 20 000	**810 008 576**	**24.6%**

Table 3: Examples of extracted genetic subsequences

V-grams	Frequency
AGCCTGGGTGACAGAGCAAGACTCTGTCTCAAAAA	118
AAGAACAAAGCTGGAGGCATCATGCTACCTGACTTCAAACTA	60
ACAGAGTTGAACCTTGCTTTCATAGTTCAGCTTTCAAACACTCTTT	30
AGTAATGGGATGGCTGGGTCAAATGGTATTTCTAGTTCTAGAT	256

increasing of the feature set size would only increase computational cost.

It is essential that experiments using the proposed method provide compression rate below 25%[8]. Based upon gzip and bzip2 compression rate, this genetic data might seems incompressible. Indeed, with the alphabet consisting mostly of 4 symbols $\{A, T, C, G\}$ with rare end-of-line separators, compression rate above 25% is not successful. With the proposed method applied, results below 25% appear, which shows that the examined data is compressible indeed.

Another significant characteristic of the proposed method is an unlimited length of v-grams, which is illustrated by the dictionary elements produced for this compression task. Examples of nontrivial dictionary elements of long length with frequencies are presented in Tab. 3. These examples are called nontrivial in contrast to trivial sequences like homopolymers or simple repetitions like *"ATAT..."*. It is worth noticing that observed frequencies are much greater than expected frequencies of random sequences ($29 \gg \frac{3 \cdot 10^{10}}{4^{47}} \simeq 10^{-17}$). It is fair to assume that such sequences might be overrepresented motifs (like cis-regulatory elements) and serve the interests of biologists.

URL compression. The proposed method can be successfully applied to a variety of tasks associated with large lists of URLs, such as clickstream data utilized by web search engines and many others (geotracks, network events, etc.). The effectiveness of the method in an application for these tasks is based on the repeating sequences, which often occurs in URL data. An experiment on digital object identifier (DOI) URLs list[9] compression is provided. This dataset consists of about 50 million journal article DOIs from CrossRef's OAI-PMH server. Results presented in Tab. 4 were calculated for DOI URLs 2013.

It is worth mentioning that a dictionary constructed in the process of compression might be used as a feature set. This property

[8]25% correspond to a 2-bit representation of each ACTG term
[9]https://archive.org/details/doi-urls

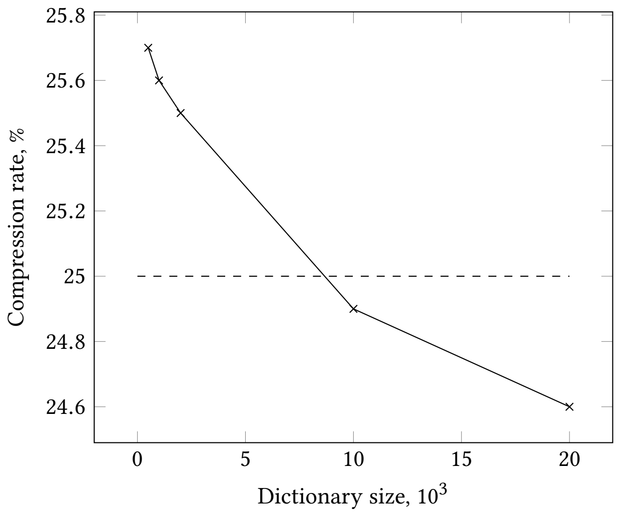

Figure 1: Genetic data compression rate depending on the size of dictionary

Table 4: DOI URLs 2013 as a stream compression

Method	Size, bytes	Compression rate, %
Original data	933 550 241	-
bzip2	154 035 790	16.5%
Dictonary size 500 000	**112 026 029**	**12.0%**

offers the challenge to utilize the method for malicious URL detection and other information retrieval tasks. Experiments with dictionary elements acting as a feature set for a text classification task are as follows.

4.3 Text classification

In this section, the application of the proposed method to feature extraction is examined. Given a textual data, a dictionary of v-grams is constructed and can be used instead of words for text classification. Performance is compared with other published results on popular datasets, including method, proposed in [21] as far as it based on the same MDL principle.

We have normalized the text data in the following way: all symbols, except letters and digits, were excluded from the text, then the text was converted to lower case. The resulting "character soup" string was passed to the method. If the method can demonstrate competitive with NLP based techniques, we can expect it to be relevant on other data types.

The proposed method is unsupervised, which allows constructing a dictionary on the whole collection of documents including the unlabeled part. Another useful property is that the method has the dictionary size parameter and unlike other methods will not increase the size of the dictionary to some unreasonable value. The other function of this parameter is to control overfit of the model. During the experiments, we studied the influence of this parameter on the results.

Table 5: Text classification accuracy on 20 NewsGroups dataset

Method	Accuracy
Discriminative RBM [17]	76.2
Naive Bayes [18]	81.8
Compressed FL [21]	83.0
Bag-of-Words SVM	83.5
v-grams SVM	82.5
v-grams + BoW SVM	**89.5**

Table 6: Example v-grams extracted from 20 NewsGroups

V-grams	Frequency
bretttheres.(74 more chars here).anocase	16
bontchevfbihhinformatikunihamburgdeorganization	18
bookofmormon	53
brownbatmanbmdtrwcom	42
comprehensivegeneralunifiedtheory	13
iamlookingfor	161
icanthinkof	70
awarethat	82
artillery	48
ator	570

20 Newsgroups. The 20 Newsgroups dataset[10] is a collection of 20 classes of different newsgroups, containing approximately 20 000 documents [16]. For this dataset, a dictionary consisting of 15 000 elements is constructed, and the SVM algorithm with the linear kernel as a classifier is used. Experiments are conducted varying size of the dictionary, and the best results were achieved with 15 000 elements in the dictionary. Sklearn implementation of the text pipeline is used as a bag-of-word baseline. Results are presented in the Tab. 5. Please note that the result of the combined feature set is hard to beat. We explain this result by the fact that the combination of unique words and informative v-grams gives a much more complete view of the textual information than any single source. Unigram dictionary has 130116 entries, many of them have a low frequency in the collection and thus cannot be extracted by statistical methods, on the other hand in a broader context these words are used and bring information to the reader. V-grams dictionary is much smaller, but it brings a statistical perspective on the data.

It is easy to note, that results of the v-grams alone are competitive and allow us to claim that v-gram based dictionary gives an ability to the researchers to use bag-of-word and other word-based techniques on the data that has no word partitioning. From the practical perspective, v-grams has at least two significant advantages over words: straightforward text normalization and fixed dictionary volume (in this case it was >8 times smaller).

To demonstrate properties of statistical view on textual data we have presented several items from the dictionary in Tab. 6. As could be seen from these examples v-grams can be quite long. It is essential that such v-grams not be examples of the overfit: the

[10]http://qwone.com/~jason/20Newsgroups/

Table 7: Named entity recognition from 20 NewsGroups

V-grams	Frequency
universityofpittsburgh	46
universityofsoutherncalifornialosangelesca	68
universityoftennesseecomputingcenter	18
universityoftennesseedivisionofcontinuingeducation	30
universityoftorontochemistrydepartment	52
universityofvirginialines	121
universityofwashington	49
universityofwashingtonseattle	54
universityofwaterloo	43
universityofwestminster	19
universityofwisconsineauclairelines	33

first one is the author's motto, and the second one is organization name (part of newsgroup letter format). The occurrence of such long strings as dictionary element reflects the fact that these strings are very informative in the sense of compression and often appear as a pattern. Another interesting effect involves automatic entity extraction. According to 20 Newsgroups format many messages are annotated with the organization corresponding to the author. These organizations were successfully mined from the dataset by the proposed algorithm. Several examples are demonstrated in Tab. 7[11]. The third type of entries in the dictionary are idioms and word+preposition pairs that bring more information than words along. The fourth type is plain words. The last type of observed data are subwords that have proven [3] their value for the text analysis.

ACL IMDb movie review. Another text collection is examined, to demonstrate the persistence of algorithm performance on classification task. Namely ACL IMDb movie review dataset[12]. This collection consists of 25 000 highly polar movie reviews for training and 25 000 for testing [19]. The size of the dictionary constructed by the proposed method is 15 000 and applied classifier is SVM. The evaluation with the comparison to other methods is presented in Tab. 9. As one can see in Tab. 8 the types of extracted items remain the same. Although v-grams specific for the collection like "5outof10" were extracted.

We have also tested if it is possible to transfer v-gram dictionary from one collection to another. For this experiment, we have used a dictionary built on 20newsgroup for IMDb classification. The drop in classification quality was significant but results are still competitive, and the combination of 20newsgroup v-grams and unigrams gives second best result in our testing. So it is possible to learn v-gram dictionary on one set of data of specific type then and effectively use it on another.

5 CONCLUSIONS

In this paper, a new method of feature extraction from sequential data, based on the minimum description length principle, is presented. MDL principle is implemented from the information theory perspective, utilizing a dictionary coder for this purpose. A feature set, constructed by the proposed method, is an optimal feature set in contrast with the output of feature selection methods. Moreover,

[11]All shown entries are preceded by "organization" prefix.
[12]http://ai.stanford.edu/~amaas/data/sentiment/

Table 8: Example v-grams extracted from IMDb dataset

V-grams	Frequency
0outof10	12
10outof10[a]	67
allofasudden	35
alltheactors	45
crouchingtigerhiddendragon	10
denzelwashington	23
despitethefactthat	50
backandforth	47
badacting	154

[a]All similar v-grams were extracted

Table 9: Text classification accuracy on IMDb dataset

Method	Accuracy
Bag-of-Words SVM [15, 19]	88.2
Word Vectors [19]	88.9
Compressed FL [21]	90.4
v-grams SVM	88.8
v-grams + BoW SVM	**94.5**
v-grams (20news)	86.8
v-grams (20news) + BoW SVM	**94.0**

feature selection methods are often of supervised nature, while the proposed method is unsupervised, which serves as a big virtue.

Another significant characteristic of the proposed method is that constructed v-grams are not limited in length, which allows extracting effective v-grams no matter how long they are. This property is essential for both compression and classification tasks. For compression task, the use of v-grams provides a better ratio. For classification task, it provides automatic entity extraction, idioms utilization and other features discovered during experiments.

In a series of experiments, main characteristics of the method and its effectiveness are demonstrated. A method can be successfully applied for compression of any sequential stream data. Performance in compression experiments surpasses popular compression algorithms, such as gzip and bzip2. What is more, after dictionary construction compression of data is exceptionally cost-efficient regarding computational overhead. On top of that, elements of dictionary constructed in compression process can be utilized as useful features, thus solving two tasks at once: data compression and feature extraction.

For text classification, a quality comparison of automatically extracted "terms" vs. well-known bag-of-words model is provided. The classification results demonstrated by the extracted feature set were competitive with the NLP based techniques. If we combine the v-grams with the words, the results become much better then on any single feature set and provide a new strong baseline. We have demonstrated that v-grams built on one dataset could work on the other of the same type. This fact opens an opportunity for further transfer learning research.

It is worth mentioning that the proposed method is 100% pure statistics based and unrelated to natural language preprocessing. Obtained within this framework results are compared to results provided by the state-of-the-art methods with natural language processing. This offers the challenge to deal with data, for which preprocessing is either difficult (e.g., Oriental languages) or irrelevant (e.g., DNA data).

Only a small number of possible v-grams applications was examined in this paper. The most natural directions of the future work: v-grams embedding, application to Oriental languages processing, preprocessing of sequential data before neural networks or graphical models training. In general, v-grams provides a tool to control the optimal size of the alphabet in sequential data processing: either use bitwise coding to reduce the alphabet or proceed with v-gram learning to expand.

Another important direction for future work is the application of v-grams to data analysis, extending beyond data compression and text classification. As demonstrated in experimental section 4, v-grams provide insights into data and offer the challenge to discoveries.

REFERENCES

[1] Francis R. Bach. 2011. Learning with Submodular Functions: A Convex Optimization Perspective. *CoRR* abs/1111.6453 (2011). arXiv:1111.6453 http://arxiv.org/abs/1111.6453

[2] Dario Benedetto, Emanuele Caglioti, and Vittorio Loreto. 2002. Language trees and zipping. *Physical Review Letters* 88, 4 (2002), 487021–487024. https://doi.org/10.1103/PhysRevLett.88.048702

[3] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. (2016). arXiv:1607.04606 http://arxiv.org/abs/1607.04606

[4] Andrej Bratko, Bogdan Filipič, Gordon V. Cormack, Thomas R. Lynam, and Blaž Zupan. 2006. Spam filtering using statistical data compression models. *Journal of Machine Learning Research* 7 (2006), 2673–2698.

[5] Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján. 2012. Conditional likelihood maximisation: A unifying framework for information theoretic feature selection. *Journal of Machine Learning Research* 13 (2012), 27–66.

[6] Rudi Cilibrasi and Paul M. B. Vitányi. 2005. Clustering by compression. *IEEE Transactions on Information Theory* 51, 4 (April 2005), 1523–1545. https://doi.org/10.1109/TIT.2005.844059

[7] David Pereira. Coutinho and Mário A. T. Figueiredo. 2015. Text Classification Using Compression-Based Dissimilarity Measures. *International Journal of Pattern Recognition and Artificial Intelligence* 29, 5, Article 1553004 (2015). https://doi.org/10.1142/S0218001415530043

[8] Eibe Frank, Chang Chui, and Ian H. Witten. 2000. Text categorization using compression models. In *Data Compression Conference Proceedings*. 555. https://doi.org/10.1109/DCC.2000.838202

[9] Shuyang Gao, Greg Ver Steeg, and Aram Galstyan. 2016. Variational Information Maximization for Feature Selection. In *NIPS*. 487–495.

[10] Isabelle Guyon and André Elisseeff. 2003. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research* 3 (2003), 1157–1182.

[11] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lotfi A. Zadeh. 2006. *Feature Extraction: Foundations and Applications (Studies in Fuzziness and Soft Computing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[12] Miten Jain, Sergey Koren, Josh Quick, Arthur C. Rand, Thomas A. Sasani, John R. Tyson, Andrew D. Beggs, Alexander T. Dilthey, Ian T. Fiddes, Sunir Malla, Hannah Marriott, Karen H. Miga, Tom Nieto, Justin O'Grady, Hugh E. Olsen, Brent S. Pedersen, Arang Rhie, Hollian Richardson, Aaron Quinlan, Terrance P. Snutch, Louise Tee, Benedict Paten, Adam M. Phillippy, Jared T. Simpson, Nicholas J. Loman, and Matthew Loose. 2017. Nanopore sequencing and assembly of a human genome with ultra-long reads. *bioRxiv* (April 2017), 128835. https://doi.org/10.1101/128835

[13] Thorsten Joachims. 1998. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *ECML (Lecture Notes in Computer Science)*, Vol. 1398. Springer, 137–142.

[14] Eamonn J. Keogh, Stefano Lonardi, and Chotirat (Ann) Ratanamahatana. 2004. Towards parameter-free data mining. In *KDD*. ACM, 206–215.

[15] Man Lan, Chew Lim Tan, and Hwee-Boon Low. 2006. Proposing a New Term Weighting Scheme for Text Categorization. In *AAAI*. AAAI Press, 763–768.

[16] Ken Lang. 1995. NewsWeeder: Learning to Filter Netnews. In *ICML*. Morgan Kaufmann, 331–339.

[17] Hugo Larochelle and Yoshua Bengio. 2008. Classification using discriminative restricted Boltzmann machines. In *ICML (ACM International Conference Proceeding Series)*, Vol. 307. ACM, 536–543.

[18] Baoli Li and Carl Vogel. 2010. Improving Multiclass Text Classification with Error-Correcting Output Coding and Sub-class Partitions. In *Canadian Conference on AI (Lecture Notes in Computer Science)*, Vol. 6085. Springer, 4–15.

[19] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *ACL*. The Association for Computer Linguistics, 142–150.

[20] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval.* Cambridge University Press.

[21] Hristo S. Paskov, Robert West, John C. Mitchell, and Trevor J. Hastie. 2013. Compressive Feature Learning. In *NIPS*. 2931–2939.

[22] Jorma Rissanen. 1978. Modeling by shortest data description. *Automatica* 14, 5 (1978), 465–471.

[23] David Sculley and Carla E. Brodley. 2006. Compression and Machine Learning: A New Perspective on Feature Space Vectors. In *DCC*. IEEE Computer Society, 332–332.

[24] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *ACL (1)*. The Association for Computer Linguistics.

[25] Henning Sperr, Jan Niehues, and Alex Waibel. 2013. Letter N-Gram-based Input Encoding for Continuous Space Language Models. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality.* Association for Computational Linguistics, 30–39.

[26] Bo Tang, Steven Kay, and Haibo He. 2016. Toward Optimal Feature Selection in Naive Bayes for Text Categorization. *IEEE Trans. Knowl. Data Eng.* 28, 9 (2016), 2508–2521.

[27] Alper Kursat Uysal and Serkan Günal. 2012. A novel probabilistic feature selection method for text classification. *Knowl.-Based Syst.* 36 (2012), 226–235.

[28] Jorge R. Vergara and Pablo A. Estévez. 2014. A review of feature selection methods based on mutual information. *Neural Computing and Applications* 24, 1 (2014), 175–186.

[29] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *NIPS*. 649–657.

[30] Jacob Ziv and Abraham Lempel. 1977. A universal algorithm for sequential data compression. *IEEE Trans. Information Theory* 23, 3 (1977), 337–343.