Journal of Machine Learning Research 13 (2012) 3247-3251 Submitted 5/12; Revised 8/12; Published 11/12

Sally: A Tool for Embedding Strings in Vector Spaces

Konrad Rieck KONRAD.RIECK@UNI-GOETTINGEN.DE
University of Göttingen
Goldschmidtstraße 7
37077 Göttingen, Germany

Christian Wressnegger WRESSNEGGER@IDALAB.DE
idalab GmbH
Adalbertstraße 20
10977 Berlin, Germany

Alexander Bikadorov ABIKU@CS.TU-BERLIN.DE
Technische Universität Berlin
Franklinstraße 28/29
10587 Berlin, Germany

Editor: Antti Honkela

Abstract

Strings and sequences are ubiquitous in many areas of data analysis. However, only few learning methods can be directly applied to this form of data. We present Sally, a tool for embedding strings in vector spaces that allows for applying a wide range of learning methods to string data. Sally implements a generalized form of the bag-of-words model, where strings are mapped to a vector space that is spanned by a set of string features, such as words or n-grams of words. The implementation of Sally builds on efficient string algorithms and enables processing millions of strings and features. The tool supports several data formats and is capable of interfacing with common learning environments, such as Weka, Shogun, Matlab, or Pylab. Sally has been successfully applied for learning with natural language text, DNA sequences and monitored program behavior.

Keywords: string embedding, bag-of-words models, learning with sequential data

1. Introduction

Strings and sequences are a common representation of data and many applications of machine learning center on analyzing strings, for example, for discovering topics in natural language text, identifying genes in DNA, or filtering spam messages. However, the vast majority of learning methods can not be directly applied to string data, as these methods usually operate in vector spaces and require numerical vectors as input for learning. While a large body of research has studied techniques for learning with strings—most notably work on mapping strings to vectors (see Salton et al., 1975; Damashek, 1995) and string kernels (see Sonnenburg et al., 2007; Shawe-Taylor and Cristianini, 2004)—only few software tools have been made available to the community so far.

In this article, we present Sally, a general-purpose tool for mapping a set of strings to a set of vectors. This mapping is referred to as *embedding of strings* and allows for applying a wide range of learning methods to string data. Sally implements a generalized form of the bag-of-words model, where strings are mapped to a high-dimensional vector space that is spanned by a set of string features. Different types of features are supported for this embedding, which range from words

delimited by whitespace characters to positional and sorted n-grams. Sally proceeds by counting the occurrences of these features in each string and generating a sparse vector of frequency values. The implementation of Sally builds on string algorithms with linear run-time and space complexity, which enables processing millions of strings and features.

Sally is not the only tool for learning with strings; some learning toolboxes also provide support for extracting features from strings or computing string kernels. The respective implementations are often tightly coupled with the toolboxes and restricted to specific applications. By contrast, Sally can be thought of as a "Swiss Army Knife" for embedding strings: Instead of targeting a single toolbox or application, Sally provides a generic link between string data and learning methods. The tool supports several data formats and is capable of interfacing with common learning environments, such as Weka, Shogun, Matlab, or Pylab. Sally has been successfully applied in diverse learning settings, including text categorization, intrusion detection, clustering of malicious software, and analysis of electricity consumption (cf. Rieck and Laskov, 2008; Wahl et al., 2009; Rieck et al., 2011; Jawurek et al., 2011).

2. Embedding of Strings

Sally implements a generalized form of the classic bag-of-words model (Salton et al., 1975). A string is represented by a set of string features ("the bag") and mapped to a vector space whose dimensions are associated with the occurrences of these features. This association is created using a hash function, where the hash value of each feature defines its dimension. Moreover, the tool extends the original bag-of-words model by supporting features derived from string kernels, such as the spectrum kernel (Leslie et al., 2002), the weighted-degree kernel (Sonnenburg et al., 2007) and the word kernel (Lodhi et al., 2002).

2.1 String Features

Sally supports three basic types of string features for constructing a bag-of-words model. These types are defined implicitly by specifying delimiters (Configuration: `ngram_delim`) and the number of consecutive bytes/words to consider (Configuration: `ngram_len`).

1. *Words.* The strings are partitioned into substrings using a set of delimiter characters D. Such partitioning is typical for natural language processing, where the delimiters are usually defined as whitespace and punctuation characters (Configuration: `ngram_delim =` D`;` and `ngram_len = 1;`).

2. *Byte n-grams.* The strings are characterized by overlapping byte sequences of length n. These features are frequently used, if only little information about the structure of the strings is known, such as in bioinformatics and computer security (Configuration: `ngram_delim = "";` and `ngram_len =` n`;`).

3. *Word n-grams.* The strings are described by overlapping word sequences of length n. These features require the definition of delimiters D and a length n. They are often used as a coarse way for capturing structure in text and tokens (Configuration: `ngram_delim =` D`;` and `ngram_len =` n`;`).

Additionally to these basic types, Sally supports extensions for refining the set of string features. For instance, inspired by the weighted-degree kernel (Sonnenburg et al., 2007), Sally supports the

extraction of *positional features*. Each string feature is extracted along with its position in the string (Configuration: `ngram_pos = 1;`). As an example, the string `abab` then contains the positional 2-grams $\texttt{ab}_1$, $\texttt{ba}_2$ and $\texttt{ab}_3$.

Moreover, as with any data, strings can suffer from noise. Words may be swapped in text and DNA bases flip positions due to mutations. Such noise can be compensated by the extension of *sorted n-grams* (Configuration: `ngram_sort = 1;`). After the extraction of an n-gram, its elements are sorted, thereby removing the local ordering of the data. This removal may improve performance, if the strings suffer from local perturbations.

Finally, Sally also supports the use of stop words (Configuration: `stopword_file`) and the thresholding of values (Configuration: `thres_low` and `thres_high`). When analyzing natural language text, these extensions allow for filtering irrelevant and (in)frequent words from the resulting feature vectors.

2.2 Embedding Function

After the extraction of string features, Sally proceeds to construct a feature vector for each string in the defined output format. This construction can be formally defined as a mapping function Φ (see Rieck and Laskov, 2008),

$$\Phi : \mathcal{X} \longrightarrow \mathbb{R}^{|\mathcal{S}|}, \quad \Phi : x \longmapsto (\phi_s(x))_{s \in \mathcal{S}},$$

where $\mathcal{X}$ corresponds to the domain of strings and $\mathcal{S}$ to the set of (hashed) string features. Depending on the configuration, the inner function ϕ either returns the number of occurrences of the feature s in the string x, a binary flag for the occurrence of s or an TFIDF weighting of s. Furthermore, different normalizations can be applied to the resulting vectors (Configuration: `vect_embed =` *embed* and `vect_norm =` *norm*).

The size of the vector space can be controlled using the number of bits for hashing the string features (Configuration: `hash_bits`), where for k bits the vector space may span up to 2^k dimensions. While such high dimensionality is favorable for constructing an expressive representation of strings, it requires an efficient implementation to guarantee a tractable run-time. Fortunately, the number of features extracted by Sally is linear in length of each string and the resulting vectors are extremely sparse. This sparsity enables Sally to embed the strings in linear time and space, irrespective of the dimensionality of the vector space (cf. Rieck and Laskov, 2008; Shi et al., 2009).

3. Run-time Evaluation

To illustrate the efficient implementation of Sally, we conduct a run-time evaluation with data sets from four application domains: DNA sequences (ARTS; Sonnenburg et al., 2006), protein sequences (SPROT; O'Donovan et al., 2002), email messages (ENRON; Klimt and Yang, 2004) and text documents (RFC; www.ietf.org/rfc.html). Statistics of the data sets are shown in Table 1. As a baseline, we consider a typical Matlab and Python script for embedding strings. Both scripts are 60–70 lines long and make use of hashing for efficiently mapping strings to vectors. For each data set and implementation, we measure the run-time for embedding strings using byte/word 5-grams as features. We set the size of the feature hashing to 24 bits ($\approx$ 16.7 million dimensions). The evaluation is conducted on an Intel Xeon CPU with 2.6GHz with 4 GB of memory.

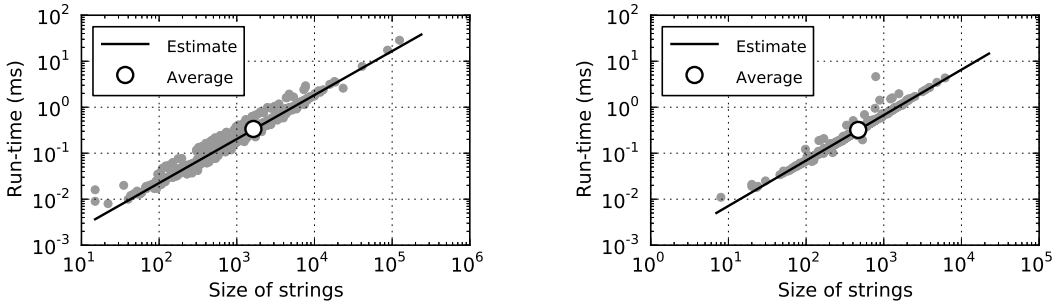
Results for the evaluation are shown in Table 1. All implementations are able to embed the strings in reasonable time (less than 10 minutes). However, Sally consistently outperforms the

other implementations. In comparison with the Python script, Sally embeds the strings 2.5× faster on average, where for Matlab a speedup of 9.5× is attained. This performance demonstrates the efficiency of Sally on different types of data, rendering it the tool of choice in applications where time matters, such as in large-scale and on-line learning.

Data sets	**ARTS**	**SPROT**	**ENRON**	**RFC**
Data set size	10^8 DNA bases	10^7 proteins	10^6 words	10^7 words
Number of strings	46,794	150,807	33,702	4,590
String features	byte 5-grams	byte 5-grams	word 5-grams	word 5-grams
Number of features	1,024 ($=4^5$)	2,800,728	4,070,853	12,136,059
Run-time performance				
Matlab script	528 s (9.6×)	381 s (7.3×)	158 s (11.4×)	530 s (9.6×)
Python script	140 s (2.5×)	183 s (3.5×)	30 s (2.1×)	113 s (2.1×)
Sally	55 s —	52 s —	14 s —	55 s —

Table 1: Run-time performance of Sally and typical scripts for embedding strings.

A fine-grained analysis of the run-time of Sally is shown in Figure 1. The embedding shows a linear run-time complexity even for large strings. On average, Sally is able to map a string to a vector within 0.3 ms including reading and writing of data, which amounts to an overall throughput of 3,000 strings per second.



(a) Email messages (word 5-grams)

(b) Protein sequences (byte 5-grams)

Figure 1: Detailed run-time analysis of Sally. The run-time per string is measured on the email messages of ENRON and the protein sequences of SPROT.

4. Conclusions

Sally provides a generic link between the wealth of string data and the available machinery of learning methods. Instead of targeting a specific application domain, the tool implements a generalized form of the bag-of-words model, which allows for learning with various types of string data, such as natural language text (Rieck and Laskov, 2008), payloads of network packets (Wahl et al., 2009) or even traces of electricity consumption (Jawurek et al., 2011). Moreover, by supporting different input and output formats, the tool can easily interface with common learning environments. Sally is open source software and available at the webpage *http://mlsec.org/sally*.

References

M. Damashek. Gauging similarity with *n*-grams: language-independent categorization of text. *Science*, 267(5199):843–848, 1995.

M. Jawurek, M. Johns, and K. Rieck. Smart metering de-pseudonymization. In *Proc. of Annual Computer Security Applications Conference (ACSAC)*, pages 227–236, Dec. 2011.

B. Klimt and Y. Yang. The Enron corpus: a new dataset for email classification research. In *Proc. of Conference on Email and Anti-Spam (CEAS)*, 2004.

C. Leslie, E. Eskin, and W. Noble. The spectrum kernel: a string kernel for SVM protein classification. In *Proc. of Pacific Symposium on Biocomputing (PSB)*, pages 564–575, 2002.

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.

C. O'Donovan, M. Martin, A. Gattiker, E. Gasteiger, A. Bairoch, and R. Apweiler. High-quality protein knowledge resource: SWISS-PROT and TrEMBL. *Briefings in Bioinformatics*, 3(3): 275–284, 2002.

K. Rieck and P. Laskov. Linear-time computation of similarity measures for sequential data. *Journal of Machine Learning Research*, 9(Jan):23–48, Jan. 2008.

K. Rieck, P. Trinius, C. Willems, and T. Holz. Automatic analysis of malware behavior using machine learning. *Journal of Computer Security (JCS)*, 19(4):639–668, June 2011.

G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

Q. Shi, J. Petterson, G. Dror, J. Langford, A. Smola, and S. Vishwanathan. Hash kernels for structured data. *Journal of Machine Learning Research*, 10(Nov):2615–2637, 2009.

S. Sonnenburg, A. Zien, and G. Rätsch. ARTS: accurate recognition of transcription starts in human. *Bioinformatics*, 22(14):e472–e480, 2006.

S. Sonnenburg, G. Rätsch, and K. Rieck. Large scale learning with string kernels. In *Large Scale Kernel Machines*, pages 73–103. MIT Press, Sept. 2007.

S. Wahl, K. Rieck, P. Laskov, P. Domschitz, and K.-R. Müller. Securing IMS against novel threats. *Bell Labs Technical Journal*, 14(1):243–257, May 2009.