Character based String Kernels for Bio-Entity Relation Detection

Ritambhara Singh
Department of Computer Science
University of Virginia
Charlottesville
rs3zz@virginia.edu

Yanjun Qi
Department of Computer Science
University of Virginia
Charlottesville
yanjun@virginia.edu

Abstract

Extracting bio-entity relations has emerged as an important task due to the ever-growing number of bio-medical documents. In this paper, we present a simple and novel representation for extracting bio-entity relationships. The state-of-the-art systems for such tasks rely on word based representations and variations of linguistic driven features. In contrast, we model bio-text by the most basic character based string representation with a family of string kernels. This eliminates time consuming parsing, issue of rare words and domain specific pre-processing. This simple representation makes our approach fast and flexible for any bio-NLP dataset. We demonstrate comparable performance and faster computation time of our approach versus previous state-of-the-art kernel methods.

1 Introduction

Relation extraction from biomedical documents is an important task in knowledge representation and inference. It helps to construct and enhance structured knowledge-bases and in turn support automatic question answering and decision making. In today's era of vast amount of information collection and retrieval, the task of naming and identifying the relations between annotated bio-entities can become complex and time consuming. This can be deducted from the fact that the MEDLINE database has more than 22 million journal articles related to biomedicine. Many state-of-the art methods have been applied for the popular tasks of extracting protein-protein interaction (PPI) and drug-drug interaction (DDI) as a part of BioCreative shared task challenges (Segura Bedmar et al., 2011; Segura Bedmar et al., 2013; Krallinger et al., 2008). While these methods have achieved good performance, they mostly rely on word-level features, are dependent on time-consuming parsers or require domain knowledge for pre-processing.

This paper uses characters instead of words for bio-entity relation extraction. Characters are the most fundamental building blocks in any language. We propose to model bio-text using its most basic character-based string representation. Through a string kernel implementation, in the framework of support vector machine (SVM), we separate positive and negative interaction instances to detect bio-entity relationships. This basic representation is independent of parsers, does not require domain-related pre-processing and eliminates the rare words problem. It not only performs comparable to other state-of-the-art methods but also provides an exploration of new and simple feature sets (complementary to existing features) that have not been previously studied for bio-NLP shared tasks.

2 Related Work

Convolution-based kernel methods have been used extensively in the tasks of PPI and DDI extraction, and differ in the feature sets they explore. While the shallow linguistic (SL) kernel (Giuliano et al., 2006) uses simple linguistic features, others utilize more complex features. Constituent parse tree-based kernels, like subtree (ST) (Vishwanathan et al., 2004), subset tree (SST) (Collins and Duffy, 2001), partial tree (PT) (Moschitti, 2006) kernels, and spectrum tree (SpT) (Kuboyama et al., 2007) kernel, use subtree forms or path structures from constituent parse trees. Another category of methods use dependency parse tree-based features. This includes edit distance and cosine similarity kernels (using shortest paths) (Erkan et al., 2007), k-band shortest path spectrum (kBSPS) (Tikk et al., 2010) (a

Proceedings of the 15th Workshop on Biomedical Natural Language Processing, pages 66–71,
Berlin, Germany, August 12, 2016.

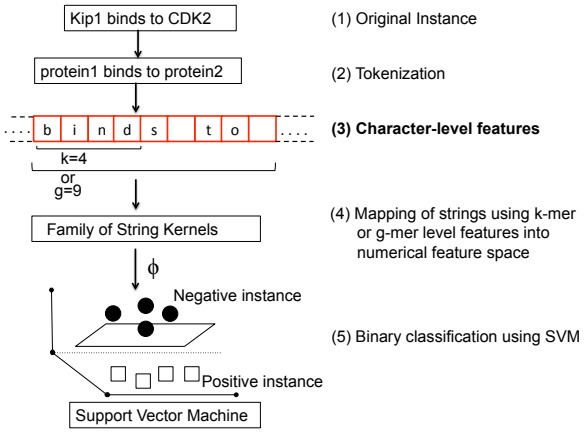

Figure 1: **End-to-end implementation of character-based string kernels for bio-entity relation detection.**

k-band extension of shortest paths), all-path graph (APG) kernel (Airola et al., 2008) (weighing differently shortest paths), and Kim's kernels (Kim et al., 2008) (combines shortest path with different lexical, part-of-speech and syntactic features). Benchmark papers, such as (Tikk et al., 2010) and (Tikk et al., 2013), have performed thorough comparative and error analyses of all these different kernels. They concluded that APG, kBSPS and SL kernels give the best performance. Therefore, we use these three kernels as baselines in our experimental comparisons. Some studies include a combination of kernels and parsers for PPI extraction task, e.g. (Miwa et al., 2009). Similarly, (Thomas et al., 2013) implemented a two-step approach to first detect general DDIs and then classify detected DDIs into subtypes. For the general DDI task, they used voting to combine kernels including APG, subtree (ST), SST, SpT, and SL kernels.

All the above discussed methods suffer from the rare words problem, and require time consuming and domain specific pre-processing steps like parsing to obtain lexical features, constituent and dependency trees. Several recent studies have discovered that character-based representation provides simple and powerful models for sentiment classification (Zheng et al., 2015) and transition-based parsing (Ballesteros et al., 2015). (Lodhi et al., 2002) first used string kernels with character level features for text categorization. However, their kernel computation used dynamic programming which is computationally intensive. Over recent years, more efficient string kernel methods have been devised (Leslie and Kuang, 2004; Kuksa et al., 2009). Therefore, we apply a family of state-of-the-art string kernels using simple character-based string representation for bio-entity relation detection in this work.

Corpus	Task	Sent.	Pos	Neg	Total
MEDLINE	DDI	1301	232	1555	1787
AIMed	PPI	1955	1000	4834	5834
LLL	PPI	77	164	166	330

Table 1: Statistics (number of sentences, positive, negative and total instances) of the MEDLINE corpus about DDI and, AIMed and LLL corpus about PPI extraction respectively.

3 Approach

Figure 1 shows our end-to-end implementation of character-based string kernel approach for bio-entity relation detection.

3.1 Character-level features

Without relying on any pre-processing, we directly use the instance sentences of bio-NLP datasets as input to the string kernels. Here, each instance (whole sentence) is viewed as one long contiguous string comprised of characters. A string kernel is then used to convert these strings into a feature space (implicitly through kernel calculation) that can be used as input for support vector machine (SVM) classification algorithm.

3.2 Family of string kernels

The key idea of string kernels is to apply a function $\phi(\cdot)$, which maps strings of arbitrary length into a vectorial feature space of fixed length. In this space, a standard classifier such as SVM (Vapnik, 1998) can then be applied. Kernel-version of SVMs calculate the decision function for an input sample x :

$$f(x) = \sum_{i=1}^{N} \alpha_i K(x_i, x) + b \qquad (1)$$

where N is the total number of training samples. String kernels (Leslie and Kuang, 2004; Kuksa et al., 2009; Ghandi et al., 2014a), implicitly compute an inner product in the mapped feature space $\phi(x)$ as:

$$K(x, x') = \langle \phi(x), \phi(x') \rangle, \qquad (2)$$

where $x = (s_1, \ldots, s_{|x|})$. $x, x' \in \mathcal{S}$. $|x|$ denotes the length of the string x. $\mathcal{S}$ represents the set of

all strings composed of dictionary Σ. $\phi : \mathcal{S} \rightarrow R^m$ defines the mapping from a sequence $x \in \mathcal{S}$ to a m-dimensional feature vector.

The feature representation $\phi(\cdot)$ plays a key role in the effectiveness of string analysis since strings cannot be readily described as feature vectors. We have implemented the following string kernels on the character representation.

Spectrum Kernel (SK): One classic representation is to represent a string as unordered set of k-mers, that is, combinations of k adjacent characters. A feature vector indexed by all k-mers records the number of occurrences of each k-mer in the current string. The string kernel using this representation is called spectrum kernel (Leslie et al., 2002), where the spectrum representation counts the occurrences of each k-mer in a string. Kernel scores between strings are then computed by taking an inner product between corresponding "k-mer - indexed" feature vectors:

$$K(x, x') = \sum_{\gamma \in \Gamma_k} c_x(\gamma) \cdot c_{x'}(\gamma) \qquad (3)$$

where γ represents a k-mer, Γ_k is the set of all possible k-mers, and $c_x(\gamma)$ is the number of occurrences (with normalization) of k-mer γ in string x. (Kuboyama et al., 2007) applied spectrum kernel on the constituent parse tree features.

Mismatch Kernel (MK): The spectrum kernel implementation is modified to include m number of mismatches in the k-mers (Leslie and Kuang, 2004; Kuksa et al., 2009). Thus, for a given k-mer γ in a string x, the (k, m)-neighborhood is generated. This consists of all k-length strings α from dictionary Σ^k such that they differ from original k-mer by at most m mismatches. The feature map of mismatch kernel can be defined as:

$$\phi_{(k,m)}(\gamma) = (\phi_\alpha(\gamma))_{\alpha \in \Sigma^k} \qquad (4)$$

where $\phi_\alpha(\gamma) = 1$ if $\alpha \in (k, m)$-neighborhood of γ, otherwise $\phi_\alpha(\gamma) = 0$. A mismatch kernel with $m = 0$ is essentially a spectrum kernel.

Wildcard Kernel (WK): For implementation of wildcard kernel, the dictionary Σ is augmented with a wildcard character $\star$ (Leslie and Kuang, 2004). Thus, the feature space consists of set of k-mers Γ_k obtained from $\Sigma \cup \{\star\}$ that consists of m occurrences of wildcard character ($\star$). The feature map of wildcard kernel can be defined as:

$$\phi_{(k,m,\lambda)}(\gamma) = \sum_{\Gamma_k inx} (\phi_\alpha(\gamma))_{\alpha \in (\Sigma \cup \{\star\})} \qquad (5)$$

where $\phi_\alpha(\gamma) = \lambda^m$ if γ matches α with m occurrences of character $\star$, otherwise $\phi_\alpha(\gamma) = 0$. Here $0 < \lambda \leq 1$.

Gapped k-mer based Kernel (GK): The previously described k-mer based string kernels generate extremely sparse feature vectors for even moderately sized values of k, resulting in overfitting. (Ghandi et al., 2014b) introduced a new feature set, called *gapped k-mers*, resolving the sparsity limitation with k-mer features. It is characterized by two parameters; (1) g, size of a gapped instance which is a segment of string including gaps and (2) k, the number of non-gapped k-mers or positions in each segment of size g. Thus, the number of gaps $d = g - k$. The inner product in equation 3 includes sum over all gapped k-mers features:

$$K(x, x') = \sum_{\gamma \in \Theta_g} c_x(\gamma) \cdot c_{x'}(\gamma) \qquad (6)$$

where γ represents a k-mer, Θ_g is the set of all possible g-mers in the given data.

3.3 Classification

Once the kernel matrix K is calculated, we input it into an SVM classifier as an empirical feature map using SVM^{light} (Joachims, 1999; Schölkopf and Burges, 1999). SVM maximizes the margin between the positive and negative instances of bio-entity interactions in the kernel defined feature space.

4 Experiment Setup

Datasets We demonstrate the benchmark implementations of our approach on three datasets with different sample sizes. They include MEDLINE corpus from the DDI extraction task (Segura Bedmar et al., 2011; Segura Bedmar et al., 2013), and the AIMed and LLL corpus from the PPI extraction task (Krallinger et al., 2008). [1]. The details of the datasets have been presented in Table 1.

Baselines We selected SL (Giuliano et al., 2006), APG (Airola et al., 2008), and kBSPS

[1]We use the same format as used in previous studies, that is, each interaction is represented as a separate input instance. Thus, a sentence about multiple interactions is represented as multiple instances. The protein name entities are replaced by special tokens. See details in (Tikk et al., 2010)

Corpus	Task	kBSPS	APG	SL	SK	(k)	MK	(k,m)	WK	(k,m)	GK	(g,k)
MEDLINE	DDI	-	82.3	78.9	82.1	(7)	82.7	(7,3)	**83**	(7,3)	82.4	(7,4)
AIMed	PPI	75.1	84.6	83.5	**75.6**	(8)	74.9	(10,5)	75.2	(10,5)	75.4	(8,6)
LLL	PPI	84.3	83.5	81.2	67.9	(7)	77.9	(7,3)	**78.4**	(8,5)	78.1	(7,5)

Table 2: Using AUC score to compare four character-based string kernels with APG, kBSPS and SL baselines. The best performing kernel parameters are also presented. AUC scores for APG and SL kernels for MEDLINE corpus have been reported in (Thomas et al., 2013), while scores of all three baseline kernels for AIMed and LLL corpus are reported in (Tikk et al., 2010).

Corpus	Task	kBSPS	APG	SL	SK	MK	WK	GK
MEDLINE	DDI	169.13	169.13	5.2	0.4	2.6	3.1	2.6
AIMed	PPI	254.15	254.14	7.82	76.8	79.5	78	41.3
LLL	PPI	10	10	0.3	0.2	1.3	1	0.2

Table 3: Comparing the kernel computation time (in seconds) for all four character-based string kernels versus the estimated parsing times of state-of-the-art baselines reported from (Tikk et al., 2010).

(Tikk et al., 2010) kernels as baselines for comparing with character-based string kernels. These kernels are the top-performing approaches, as reported by(Tikk et al., 2010; Tikk et al., 2013) and (Luo et al., 2016).

Parameters We ran all our string kernels across multiple kernel parameter settings as follows: (1) SK : $k = \{6, 7, 8, 9, 10\}$, (2) MK,WK : $k = \{6, 7, 8, 9, 10\}$ and $m = \{1, .., k-1\}$, and (3) GK : $g = \{6, 7, 8, 9, 10\}$ and $k = \{1, ..., g-1\}$. These string kernels were implemented using *gkmsvm* (Ghandi et al., 2014a) tool. The character level dictionary, $\Sigma = \{a, ..., z, 0, 1, ...9\}$ (size=36), is consistent for all the datasets and kernels.

Evaluation Metrics We performed 10-fold document level cross-validation on each selected corpus and calculated the AUC score (area under the receiver operating characteristic curve) for performance evaluation. (Tikk et al., 2010) confirmed that AUC score is more stable to parameter modifications and less sensitive to the ratio of positive/negative pairs in the corpus than F-score. Hence, AUC score is our choice for performance metric. We also recorded the kernel calculation times (in seconds) for all four string kernels.

5 Results

Table 2 summarizes the performance evaluation. The AUC scores for APG and SL kernels for MedLine corpus have been reported in (Thomas et al., 2013), while scores of all baseline kernels for AIMed and LLL corpus are reported in (Tikk et al., 2010). Our string kernel approaches, with simple character features, (WK,MK, and GK), outperform the baseline kernels (Table 2) on the MedLine corpus. For the AIMed corpus, SK, GK, and WK give higher AUC score than the baseline kBSPS kernel. Our methods give reasonable performance for LLL corpus as well, however not as good as the three baseline kernels. The parameters giving the best AUC performance are also reported (Table 2). Our representation is complementary and can be plugged into state-of-the-art baselines to further improve their systems.

Table 3 presents the kernel computation time comparison of all four character-based string kernels versus the baselines. We compare this with the estimated parsing times. For the baseline kernels, these have been reported and calculated in (Tikk et al., 2010). Unlike baseline kernels, we use character level features directly and thus do not need the parsing step.

6 Discussion

We have proposed a simple and novel character representation for bio-entity relation detection task. We implement a family of string kernels on such simple features extracted directly from instances of PPI and DDI extraction task datasets. This eliminates time-consuming and domain-specific pre-processing steps, making our approach fast and flexible for any bio-NLP dataset. Hence, our work opens new avenues to explore different and simpler feature sets at the character level.

References

Antti Airola, Sampo Pyysalo, Jari Björne, Tapio Pahikkala, Filip Ginter, and Tapio Salakoski. 2008. All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. *BMC bioinformatics*, 9(11):1.

Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. *arXiv preprint arXiv:1508.00657*.

Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Advances in neural information processing systems*, pages 625–632.

Günes Erkan, Arzucan Özgür, and Dragomir R Radev. 2007. Semi-supervised classification for extracting protein interaction sentences using dependency parsing. In *EMNLP-CoNLL*, volume 7, pages 228–237.

Mahmoud Ghandi, Dongwon Lee, Morteza Mohammad-Noori, and Michael A Beer. 2014a. Enhanced regulatory sequence prediction using gapped k-mer features. *PLoS Comput Biol*, 10(7):e1003711.

Mahmoud Ghandi, Morteza Mohammad-Noori, and Michael A Beer. 2014b. Robust k-mer frequency estimation using gapped k-mers. *Journal of mathematical biology*, 69(2):469–500.

Claudio Giuliano, Alberto Lavelli, and Lorenza Romano. 2006. Exploiting shallow linguistic information for relation extraction from biomedical literature. In *EACL*, volume 18, pages 401–408. Citeseer.

Thorsten Joachims. 1999. Making large scale SVM learning practical. Technical report, Universität Dortmund.

Seonho Kim, Juntae Yoon, and Jihoon Yang. 2008. Kernel approaches for genic interaction extraction. *Bioinformatics*, 24(1):118–126.

Martin Krallinger, Florian Leitner, Carlos Rodriguez-Penagos, Alfonso Valencia, et al. 2008. Overview of the protein-protein interaction annotation extraction task of BioCreative II. *Genome biology*, 9(Suppl 2):S4.

Tetsuji Kuboyama, Kouichi Hirata, Hisashi Kashima, Kiyoko F Aoki-Kinoshita, and Hiroshi Yasuda. 2007. A spectrum tree kernel. *Information and Media Technologies*, 2(1):292–299.

Pavel P Kuksa, Pai-Hsi Huang, and Vladimir Pavlovic. 2009. Scalable algorithms for string kernels with inexact matching. In *Advances in Neural Information Processing Systems*, pages 881–888.

Christina Leslie and Rui Kuang. 2004. Fast string kernels using inexact matching for protein sequences. *The Journal of Machine Learning Research*, 5:1435–1455.

Christina S Leslie, Eleazar Eskin, and William Stafford Noble. 2002. The spectrum kernel: A string kernel for SVM protein classification. In *Pacific symposium on biocomputing*, volume 7, pages 566–575.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444.

Yuan Luo, Özlem Uzuner, and Peter Szolovits. 2016. Bridging semantics and syntax with graph algorithms–state-of-the-art of extracting biomedical relations. *Briefings in bioinformatics*, page bbw001.

Makoto Miwa, Rune Sætre, Yusuke Miyao, and Jun'ichi Tsujii. 2009. Protein–protein interaction extraction by leveraging multiple kernels and parsers. *International journal of medical informatics*, 78(12):e39–e46.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Machine Learning: ECML 2006*, pages 318–329. Springer.

Bernhard Schölkopf and Christopher JC Burges. 1999. *Advances in kernel methods: support vector learning*. MIT press.

Isabel Segura Bedmar, Paloma Martinez, and Daniel Sánchez Cisneros. 2011. The 1st DDIextraction-2011 challenge task: Extraction of drug-drug interactions from biomedical texts.

Isabel Segura Bedmar, Paloma Martínez, and María Herrero Zazo. 2013. Semeval-2013 task 9: Extraction of drug-drug interactions from biomedical texts (ddiextraction 2013). Association for Computational Linguistics.

Philippe Thomas, Mariana Neves, Tim Rocktäschel, and Ulf Leser. 2013. WBI-DDI: drug-drug interaction extraction using majority voting. In *Second Joint Conference on Lexical and Computational Semantics (* SEM)*, volume 2, pages 628–635.

Domonkos Tikk, Philippe Thomas, Peter Palaga, Jörg Hakenberg, and Ulf Leser. 2010. A comprehensive benchmark of kernel methods to extract protein–protein interactions from literature. *PLoS Comput Biol*, 6(7):e1000837.

Domonkos Tikk, Illés Solt, Philippe Thomas, and Ulf Leser. 2013. A detailed error analysis of 13 kernel methods for protein–protein interaction extraction. *BMC bioinformatics*, 14(1):1.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience, September.

SVN Vishwanathan, Alexander Johannes Smola, et al. 2004. Fast kernels for string and tree matching. *Kernel methods in computational biology*, pages 113–130.

Xiaoqing Zheng, Haoyuan Peng, Yi Chen, Pengjing Zhang, and Wenqiang Zhang. 2015. Character-based parsing with convolutional neural network. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1054–1060. AAAI Press.