Pattern Recognition Letters 31 (2010) 1591–1597

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

A composite kernel for named entity recognition

Sujan Kumar Saha *, Shashi Narayan, Sudeshna Sarkar, Pabitra Mitra

Computer Science and Engineering Department, Indian Institute of Technology Kharagpur, Kharagpur 721302, India

ARTICLE INFO

Article history:
Received 21 August 2009
Available online 8 May 2010
Communicated by T. Vasilakos

Keywords:
Named entity recognition
Support vector machine
Kernel methods
String kernel
Machine learning

ABSTRACT

In this paper, we propose a novel kernel function for support vector machines (SVM) that can be used for sequential labeling tasks like named entity recognition (NER). Machine learning methods like support vector machines, maximum entropy, hidden Markov model and conditional random fields are the most widely used methods for implementing NER systems. The features used in machine learning algorithms for NER are mostly string based features. The proposed kernel is based on calculating a novel distance function between the string based features. In tasks like NER, the similarity between the contexts as well as the semantic similarity between the words play an important role. The goal is to capture the context and semantic information in NER like tasks. The proposed distance function makes use of certain statistics primarily derived from the training data and hierarchical clustering information. The kernel function is applied to the Hindi and biomedical NER tasks and the results are quite promising.

1. Introduction

A named entity (NE) denotes a noun or noun phrase referring to a name belonging to a predefined category like person, location and organization. Named entity recognition (NER) is the task of identifying and categorizing the named entities from text. Named entities are often the pivotal as well as the most information-bearing elements of a text, and NER systems find application in a number of tasks like information extraction, text mining and machine translation. Due to its immense importance, a substantial amount of work has been carried out for NER system development in various languages and domains.

The use of support vector machines (SVM) is quite common in NER and other natural language processing (NLP) tasks. SVM (Vapnik, 1995) is a margin based approach where the similarity between the instances, which are composed of the data along with the corresponding feature values, is used to define the classifier. The computation of the similarity (or distance) between the instances plays a very important role in the performance of a SVM classifier.

In NER task the most commonly used features are the surrounding words, suffix and prefix information. One approach of using such string features in SVM classifier is to use binary feature vectors (Kudo and Matsumoto, 2001; Isozaki and Kazawa, 2002; Takeuchi and Collier, 2002), where a particular feature is converted into several binary values. For example, the feature 'previous word' is converted into N binary features where N is the total number of unique words in the lexicon. Another approach has been to define a kernel function that is directly applicable to the string features. Several types of 'string kernels' (Leslie et al., 2002; Lodhi et al., 2002; Tian et al., 2007) have been defined based on the fact that the similarity between two particular strings depends on the number of common sub-strings they have. String kernels have been used successfully in various tasks like text classification, protein classification, and entity and relation extraction.

But we feel that the binary representation and sub-string similarity based string kernels are not able to capture well the semantic similarity between the instances in context sensitive tagging tasks like NER. For example, the two words 'Prof.' and 'Chairman' have some similarity in the context of the NER task as both of these occur at the preceding position of the person named entities. Such similarity between the words cannot be captured by binary or sub-string kernel based approaches. In order to characterize these NER task specific similarity between the features we propose a class association kernel and a hierarchical word clustering based kernel. We then form a composite kernel by combining these two kernels.

For the class association kernel, the feature space is divided into a number of sub-groups where each feature group consists of a set of similar features. The individual features are then transformed into a $c+1$ dimensional vector where c is the number of named entity classes. These vectors are based on class association based statistics derived from the training data. The similarity between the vectors in a feature group is computed by making use of an appropriate distance function. This similarity is the sub-kernel value corresponding to the feature group. Finally, the sub-kernels

* Corresponding author. Tel.: +91 9732655684; fax: +91 3222 278985.
E-mail addresses: sujan.kr.saha@gmail.com (S.K. Saha), shashi.narayan@gmail.com (S. Narayan), shudeshna@gmail.com (S. Sarkar), pabitra@gmail.com (P. Mitra).

0167-8655/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2010.05.004

are combined in a weighted fashion to obtain the final class association kernel.

In the second approach of computing the distance between the strings we use hierarchical clustering information. We have used the Brown clustering algorithm (Brown et al., 1992) to cluster the string feature values (e.g., words) based on their contextual similarity in a corpus.

The proposed kernel has been tested in NER task on two different domains. The first task we attempt is the Hindi NER task. The proposed kernel is compared with a binary feature based linear SVM classifier, substring similarity based string kernel and also with other statistical classifiers like maximum entropy (MaxEnt) and conditional random fields (CRF). We also test the proposed kernel in another domain, namely, the biomedical NER. In both the task the proposed kernel performs quite well.

2. Previous work

In this section, we present an overview of the research that have been carried out for developing NER systems in Hindi and biomedical domain. We also present a brief overview of the kernel based approaches in NER and other text processing tasks.

2.1. Hindi NER task

In the last few years a substantial amount of work has been carried out for developing NER systems in different languages and domains. For developing NER systems, machine learning (ML) based approaches have been mostly used. Several machine learning algorithms have been used for NER system development in various languages and domains. Hidden Markov model (HMM) (Collier et al., 2000; Shen et al., 2003; Zhou and Su, 2004; Ponomareva et al., 2007), MaxEnt (Borthwick, 1999; Lin et al., 2004; Kim and Yoon, 2007; Saha et al., 2008), CRF (Li and McCallum, 2004; Settles, 2004; Tsai et al., 2006; Leaman and Gonzalez, 2008), SVM (Kazama et al., 2002; Takeuchi and Collier, 2002; Lee et al., 2004), etc. are the most commonly used techniques.

Machine learning based methods are mostly used for Hindi NER system development too. Due to several language specific issues like, absence of capitalization, free word order, high ambiguity in Indian names and unavailability of sufficient resources, the Hindi NER task is quite difficult.

A pioneering work on Hindi NER is by Li and McCallum (2004) where they used CRF and feature induction. In their study the training corpus size was 340K words with 15,063 NEs belonging to three types, namely person, location and organization. They achieved a *f*-value of 71.5. Saha et al. (2008) used a training corpus containing 243K words to develop a Hindi NER system using MaxEnt classifier. They explored different NER features in Hindi language and studied their effectiveness. Gazetteer lists as well as identified context patterns were integrated in the system. Integrating all these approaches a *f*-value of 81.52 was achieved considering four NE classes (person, location, organization and date). In IJCNLP 2008 (International Joint Conference on Natural Language Processing, Hyderabad, India) a shared task[1] was organized on identification of NEs from texts in south and south-east Asian languages. Five languages were considered in the task which are Bengali, Hindi, Oriya, Telugu and Urdu. The task also included the identification of the nested NEs. The best result in the shared task was a *f*-value of 65.13 for Hindi where MaxEnt classifier and context rules were combined to prepare a hybrid system (Singh, 2008).

[1] More information on the shared task is available at: *http://ltrc.iiit.net/ner-ssea-08/index.cgi*.

2.2. Biomedical NER task

Biomedical NER task refers to the identification of biomedical named entities (like, protein, DNA, RNA) from (biomedical) text. Due to the presence of several difficulties (Shen et al., 2003), the performance of the biomedical NER systems is quite low compared to the general domain English NER systems. As in general domain, machine learning techniques are mostly used in biomedical NER. Several of the systems use a rule based postprocessing step, even though the core system is primarily built using machine learning algorithms.

In our experiments we have used the JNLPBA 2004 corpus (Kim et al., 2004). Here we mention a few systems developed using this corpus. A number of systems participated in JNLPBA 2004 shared task. Among these the highest accuracy was achieved by the system developed by Zhou and Su (2004) which achieved a *f*-value of 72.55. This system used HMM and SVM with some deep domain knowledge like in domain POS, name alias resolution, cascaded NE resolution, abbreviation detection, external name dictionaries. Without these domain knowledge the reported accuracy of the system was a *f*-value of 60.3. The second highest accuracy in the task was achieved by the maximum entropy Markov model (MEMM) based system developed by Finkel et al. (2004). This system used external resources (e.g., British National Corpus, large gazetteer lists, web), deeper syntactic features etc. to achieve a *f*-value of 70.06. Some other systems (Settles, 2004; Song et al., 2004) in the shared task that achieved good accuracy also used some amount of domain knowledge or external resources.

2.3. SVM and kernel in text processing tasks

SVM based classifiers have been used in various text processing tasks in the last few years. As most of the text processing tasks are required to use string features, several techniques have been adapted for handling the strings. Binary representation of the features is a common approach. Kudo and Matsumoto (2000, 2001) used SVM with binary feature representation in the English base phrase identification task. Takeuchi and Collier (2002) used SVM in the NER task with binary feature representation. They used the SVM classifier in the general domain NER task using MUC6 data and also in the molecular biology domain. For the NER task SVM was also used by Isozaki and Kazawa (2002). They proposed a few techniques like removal of unnecessary features to make the classifier efficient in terms of training time and performance.

The use of kernel functions applicable on string features is another popular approach for handling string features in SVM. A function that calculates the inner product between mapped instances in a feature space is a kernel function. A set of such functions have been proposed for handling string features, which are commonly named as 'string kernel'. String kernels calculate the similarity between the strings. One idea for getting the similarity between two strings is to find the amount of common substrings they contain – more substrings in common might refer to more similarity. A few kernels have been proposed based on this idea. Leslie et al. (2002) proposed the spectrum kernel and used it in the protein classification task. Leslie et al. (2004) proposed another string kernel for the protein classification task and named this as mismatch string kernel. Lodhi et al. (2002) proposed the string subsequence kernel which has been successfully used in the text classification task. Tian et al. (2007) proposed a light-weight string kernel based on matching subsequences with all possible lengths shared by two strings and used the kernel in the sequence data classification problem. The computation of such substring based kernels is complex, and some work has been done (e.g., Leslie and Kuang, 2004; Teo and Vishwanathan, 2006) in order to reduce the computation cost of the string kernels.

The kernel functions discussed above have been used successfully in several text processing tasks. In some text processing tasks like parts-of-speech tagging and NER, the context information plays a very important role in addition to the target word. The context information needs to be captured for yielding highly accurate classifiers. Vanschoenwinkel et al. (2005) defined a context sensitive kernel function for the NER task where the contexts are considered in terms of a sliding window and position specific weights are given for the individual terms in the window during the kernel computation.

3. Proposed kernel for NER task

The linear SVM computes the dot product between instances.

$$K(X,Y) = \phi(X) \cdot \phi(Y) \tag{1}$$

If $x_1, x_2, \ldots, x_n$ are features of X and $y_1, y_2, \ldots, y_n$ are features of Y,

$$(x_i \cdot y_i) = \begin{cases} 1, & \text{if } x_i \text{ and } y_i \text{ are same;} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

However when we are dealing with word features and other string based features, such dot product based similarity computation is not able to capture the NER task specific similarity between the strings. For example, the words 'Prof.' and 'Chairman' have some similarity in the context of the NER task as both occurs frequently at the preceding position of the person names; 'small' and 'large' are related words, both being adjectives used in similar contexts; 'town' and 'district' have similarity as both of these are common location terms and occur frequently at the surrounding positions of the location names. Such task specific similarity is important not only in word features but also in other string features like suffix, prefix and n-grams. For example, 'pur' and 'ganj' are two common suffixes of Indian place names. Although these are two different suffixes but these have some similarity. Such task specific similarity cannot be captured properly by the substring similarity based string kernels.

We have attempted to capture this task specific semantic similarity and leveraged this for computing the distance between the instances. We have computed the similarity in two different ways: one is a class association similarity based on a vector representation of the string features, and the other one is based on hierarchical clustering information. We have used these similarity functions as kernel in SVM. These individual functions are combined with a suitable weight and the combined function is also used as a composite kernel. These kernels are discussed below.

3.1. A class association kernel for NER

In shallow NLP tasks like NER, different string based features play an important role. We use these string based features in SVM by defining a class association (CA) kernel.

3.1.1. Feature group

We first divide the overall feature space (F) into a few (say, m) feature groups. For example, in the case of word features, if a word window of length five is used then five different features are considered which are w_0 (current word), w_{-1} (previous word), w_{+1} (next word), w_{-2} and w_{+2}. These five features form a feature group (F_{word}). Similarly suffix features up to length l refers to l different features, which form another feature group (F_{suf}). For each feature group a sub-kernel is defined. The final kernel is obtained by combining these sub-kernels:

$$K_{CA}(X,Y) = \sum_{j=1}^{m} \lambda_j K_j(X_{sub(j)}, Y_{sub(j)}) \tag{3}$$

In this equation K_j denotes the jth sub-kernel corresponding to the jth feature group and λ_j refers to its corresponding weight which represents the relative importance of the feature group in the feature space. As the linear combination of two kernels also give rise to a proper kernel, the combined function is also a kernel.

3.1.2. Feature vector representation

The sub-kernels use a numerical vector representation of the features. The vectors are of dimension $c+1$ corresponding to c NE classes and one for the not-name class. These vectors use class association based statistics derived from the training corpus. The vector representation is discussed here in detail in the context of word features.

Word features are used in form of a context window of length $p+q+1$, containing p previous words and q next words. Different position specific vectors are defined corresponding to the $p+q+1$ different positions. Now for each word (w) in the corpus we define a class specific weight, $Wt_C(w)$, which is assigned as the corresponding component of the $(c+1)$-dimensional vector. For a particular position, $Wt_C(w)$ is defined as

$$Wt_C(w) = \frac{\textit{No. of occurrence of } \mathrm{w} \textit{ in position } \mathrm{pos} \textit{ of a NE of class } \mathrm{C}}{\textit{Total no. of occurrence of } \mathrm{w} \textit{ in corpus}} \tag{4}$$

where *pos* denotes a particular position of the $p+q+1$ window. Feature vectors are defined for other feature groups similarly.

3.1.3. Computing the sub-kernels

We now discuss how the sub-kernels for the feature groups are computed. First, for each feature in a feature group we assign a relative importance. For example, for word feature group we assume that the current word is the most important feature and this is given a weight of 1. The weights of the other positions are taken to vary inversely with the square of the distance to the current word. Thus +1 and −1 positions combinedly receive a weight of 1/2; +2 and −2 positions get a total weight of 1/4. For affix features, we have currently considered only the affixes of the current word. Here we assign length specific weight distribution; higher length (e.g., $l = 5, 6$) affixes have more weight than the shorter affixes (e.g., $l = 1, 2$). For the other feature groups, weight distributions are assigned similarly.

Now the sub-kernels between the instances are computed. For two particular feature values, x_k and y_k corresponding to the kth feature, we compute the inner product (or, cosine similarity) between the corresponding vectors ($V1_k$ and $V2_k$). This similarity between two particular feature values is denoted by $K_{val(k)}$. Once all the $K_{val(k)}$ values for a particular feature group are obtained, these are combined using the corresponding weight distribution discussed above to get the final sub-kernel for the feature group.

3.1.4. Class association kernel – an example

In Table 1, we have explained the computation of the class association kernel with an example. Here we consider two Hindi instances where the target words are, '*bilAsapura*[2]' (Bilaspur) and '*kvAlAlumpura*' (Kualalumpur) occurring in two different contexts. We show how to evaluate the class association kernel for these. In the example, we have considered only word features (window 3) and suffix features (of length 5, 4 and 3).

The instances (X, Y) and the vectors ($V1$, $V2$) corresponding to each feature values are shown in the table. The $K_{val(k)}$ values are computed for each feature. During similarity computation, if two feature values are the same then the similarity is taken as 1 otherwise the similarity is the dot product between the vectors. The val-

[2] All the Hindi words are written using Itrans transliteration.

Table 1
Class association kernel computation example.

	Word features {-1 0 + 1}	Suffix features {5 4 3}
X	me bilAsapura jile	apura pura ura
V1	[.03.04.04.89] [0.8.2 0] [0.62 0.38]	[0.91.09 0] [0.9.09.01] [.02.09 0.89]
Y	me kvAlAlumpura shahara	mpura pura ura
V2	[.03.04.04.89] [0 1 0 0] [0.43 0.57]	[0 1 0 0] [0.9.09.01] [.02.09 0.89]
$K_{val(k)}$	1 0.80 0.49	0.91 1 1
K_{group}	1.17	1.16
K_{CA}	1.75	

ues of [1 0.80 0.49] denotes the similarity between the word feature values for previous (−1), current (0) and next (+1) positions respectively. Now we combine these $K_{val(k)}$ values of a feature group using corresponding weight distribution to obtain the K_{group} values for a feature group. Here the K_{group} value for the word feature group is 1.17 ($1 \times 0.80 + 0.25 \times 1 + 0.25 \times 0.49$) and for the suffix feature group is 1.16. These are combined using Eq. (3) to get the final kernel K_{CA}, which is 1.75. The value of λ_{word} is taken as 1 and λ_{suffix} is taken as 0.5.

3.2. Hierarchical clustering based kernel for NER

In the clustering based kernel we use cluster information of the feature values (e.g., words) as a measure of similarity between them. Cluster information has been used in different NLP tasks in the past. Several types of clustering techniques (e.g., Brown et al., 1992; Pereira et al., 1993; Ushioda, 1996; Biemann, 2006) have been proposed and used in various NLP tasks. Miller et al. (2004) used the hierarchical word clustering algorithm proposed by Brown et al. to extract binary string representation of the words which were encoded in features that are incorporated in a discriminatively trained name tagging model.

3.2.1. Clustering of words

Here we have used the bottom-up agglomerative clustering algorithm proposed by Brown et al. (1992). The input to the algorithm is a list of words to be clustered and a large raw corpus (we have used a raw corpus containing 2000K words). The output from the clustering algorithm is a binary tree, called dendrogram, in which the leaves are the words. Within this tree, each word is uniquely identified (hard clustering) by its path from the root which can be compactly represented by a bit sequence.

In Fig. 1 we have presented an example cluster. Here nine Hindi words are clustered (English meaning of the words are given in parenthesis in the figure). To group the words into a few clusters, the dendrogram can be cut at a particular level. For example, if we cut it at depth 2, we obtain three clusters, which are, [*ma.ntrI*, *pradhAnama.ntrI*, *netA*], [*shrI*, *putra*, *nivAsI*] and [*shahara*, *gAon*, *steshana*].

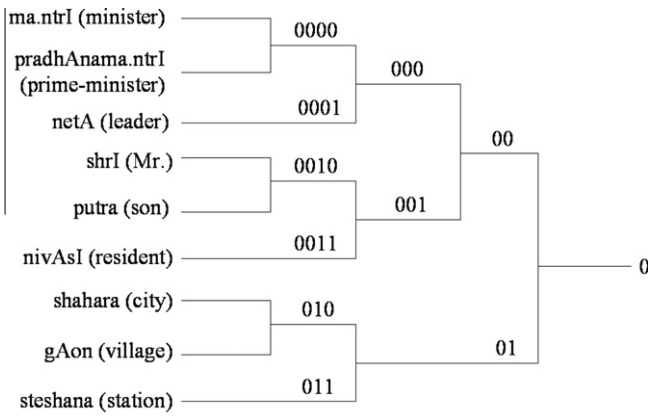

Fig. 1. An example hierarchical cluster of Hindi words.

During kernel computation to obtain the similarity between the words, we have used binary sequences. The binary sequence for each word is obtained from the dendrogram as shown in the figure. In the example, the binary sequence corresponding to *ma.ntrI* is 00000, the sequence for *putra* is 00101 and for *steshana* the sequence is 011. From the binary code sequence the similarity between two words can be obtained using the fact that, more similar words have longer prefix matching in their binary sequences. This distance is basically the dendrogram distance, the minimum number of edges required to travel from one word to the another.

3.2.2. Kernel computation

Similar to the class association kernel, we have also considered feature group and relative importance of the individual features in a feature group. The distance between two string values (e.g., words) is computed from the dendrogram using the corresponding bit sequences as explained above. Finally the individual distances are combined in a weighted fashion to obtain the kernel value of a feature group. The intra group weight we have considered here is similar to the weight distribution used in the class association kernel.

4. Experimental results: Hindi NER

This section presents the details of the experiments conducted on the Hindi NER task in order to test the effectiveness of the proposed kernel. The performance of the proposed kernel is compared with linear SVM and a substring similarity based string kernel. A comparison of performance of the proposed kernel with other machine learning classifiers like MaxEnt, CRF is also presented.

4.1. NER evaluation measures

In NER task the accuracies are measured in terms of the *f-measure*, which is the weighted harmonic mean of precision (pre) and recall (rec). *Precision* is the percentage of the correct annotations and *recall* is the percentage of the total NEs that are successfully annotated. The general expression for measuring the *f-measure* or *f-value* is,

$$F_\beta = \frac{(1+\beta^2)\ (precision \times recall)}{(\beta^2 \times precision + recall)} \tag{5}$$

Here the value of β is taken as 1.

During the evaluation we have followed the *exact match* strategy; which means a detected NE is assumed as correct if it matches exactly with the corresponding test data entity in terms of both the NE category and boundary.

4.2. Training and test data

The training data for the Hindi NER task is composed of about 200K words which is collected from the popular daily Hindi newspaper "Dainik Jagaran". Three types of names are considered, namely, *Person*, *Location* and *Organization*. The corpus has been manually annotated and contains about 5500 person, 4400 location and 2700 organization entities. The Hindi test corpus contains 25K words, which is distinct from the training corpus. The test corpus contains 678 person, 480 location and 322 organization names. To preserve the boundary information of the NEs, the corpus is

annotated using *BIO* format where *B-ne* denotes the beginning word of a NE, *I-ne* refers to the other words if the NE contains more than one word and *O* refers to the not-name words.

4.3. NER features used

NER tasks in different languages and domains require different types of features to be used. Some of the features are string features like surrounding words, suffixes up to a particular length, prefixes up to a particular length, parts-of-speech information of the current and surrounding words and some are binary features like whether the word is capitalized, whether the word contains any special character. For the Hindi NER task, a list of useful features have been explored previously by Saha et al. (2008).

In this paper, our main objective is to evaluate the performance of the proposed kernel for handling the word features and other string based features in SVM; so we focus on using a few such features. The features we use are: current word, previous three words, next three words, NE tags of the previous words, suffixes up to a length of five characters and prefixes up to a length of five characters.

4.4. Baseline hindi NER using SVM

First we build the baseline classifier using SVM where binary representation of the features is used. In binary representation a particular feature (like, word feature) is converted into several (the total number of distinct feature values) binary features. For example, our Hindi corpus contains a total of 17K distinct words, so the word feature for a particular position is transformed into 17K binary features. If a word window of length three is considered in the feature set, a total of 51K binary features are used. A linear kernel is used for preparing the baseline classifier.

For the development of the baseline system we need to select a suitable feature set. To select a suitable feature set we perform a set of experiments considering the candidate features (mentioned in Section 4.3) individually or in combination. Table 2 summarizes the results of these experiments. First we run experiments by varying the length of word window. In these experiments we have observed that when only the word features are used then a word window of length five (containing two previous and two next words) performs better. This achieves a *f*-value (Fval) of 64.88. Use of a smaller (length three) or larger (length seven) word window reduces the accuracy. When other features like suffix, prefix and previous tag information are added then the *f*-value is increased to 80.57. Interestingly this *f*-value is obtained when a word window of length three is used. When a word window of length five is used, the *f*-value becomes 80.31. This is due to overfitting, which occurs when a large number of features are applied on an insufficient training data. When the word window length is increased from three to five then a large number of ($2 \times$ total number of unique words in the corpus) binary features are added to the existing feature space (that already contains a large number of binary features) which causes overfitting. Similar overfitting was observed by Saha et al. (2008) in their MaxEnt based Hindi NER system and in Saha et al. (2009) they proposed a few feature reduction techniques to reduce overfitting. In this paper as we focus on the effectiveness of kernel functions for handling the string features directly (not converting a particular string feature into multiple binary features), we have not considered the overfitting reduction techniques.

Table 2
The baseline accuracy of different features (linear SVM with binary features).

Feature	Pre	Rec	Fval
Words – window 3	80.35	53.07	63.92
Words – window 5	81.17	54.04	64.88
Words – window 7	80.03	52.84	63.65
Words – window 5, suffix information	83.36	66.45	73.95
Words – window 5, suffix, prefix	83.62	71.2	76.91
Words – window 5, suffix, prefix, prev NE tag	86.56	75.36	**80.57**
Words – window 3, suffix, prefix, prev NE tag	86.22	75.16	80.31

4.5. Proposed kernel based Hindi NER

Now we use the proposed kernel to build the NER classifier. We have evaluated the performance of both the kernels separately and the performance of the composite kernel. We have used the kernels in two different ways to evaluate their performance; (*i*) *replacement*: replacing the baseline binary features and the linear kernel by direct string feature values and the proposed kernel and (*ii*) *merge*: adding the proposed kernel with the baseline classifier with suitable weight factor.

In Table 3 we have summarized the performance of the individual kernels and the composite kernel.

4.5.1. Performance of the class association kernel

Using the class association kernel (Section 3.1) in replacement strategy, we have achieved a *f*-value of 75.77 when only the word features are used. When these are the only features used, a SVM with the class association kernel function significantly outperforms the baseline classifier. Addition of baseline features with the class association kernel (i.e., merge strategy) increases the *f*-value to 77.2 when only word features are used.

With the addition of suffix and prefix features we become able to increase the *f*-value to 78.06 using the replacement strategy. This accuracy is obtained when the ratio among the weight of the word, suffix and prefix features is chosen as 4:1:1. When the ratio is taken as 2:1:1 the accuracy reduces to 77.59 (not shown in the table), which proves that the selection of suitable weight of the feature groups (i.e., the λ values in Eq. (3)) is important. The performance of the system is further improved when the previous tag information is added. The previous tag information is added in a binary feature representation, no vector representation is used for the previous tags.

Here we like to mention that when the kernel function is used, then a word window of length five performs better even with more features. With word window of length three the highest *f*-value is 81.52 in the class association kernel (replacement strategy) but with a window of length five the accuracy becomes 81.86. Unlike

Table 3
Performance of the proposed kernels in Hindi NER [suf: suffix, pref: prefix, ptag: previous NE tags].

Kernel	Feature	Replacement			Merge		
		Pre	Rec	Fval	Pre	Rec	Fval
CA	Words	80.16	71.84	75.77	80.81	73.72	77.2
	Word, suf, pref	82.98	73.68	78.06	82.26	76.32	79.18
	Word, suf, pref, ptag	83.39	80.38	81.86	83.56	82.76	83.21
Cluster	Words	92.24	61.02	73.45	92.5	65.04	76.38
	Word, suf, pref	91.88	67.39	77.75	90.51	69.74	78.78
	Word, suf, pref, ptag	91.17	72.92	81.03	91.94	77.63	84.18
Composite	Words	83.11	75.26	78.99	81.44	77.8	79.58
	Word, suf, pref	83.87	76.4	79.96	84.63	77.92	81.14
	Word, suf, pref, ptag	88.41	79.21	83.56	89.02	80.63	84.62

the baseline, during the kernel computation all the features are not given same weight, individual feature importance is considered here; as the long distance words (−2 and + 2) are assigned lower weight (as discussed in Section 3.1) a larger word window works better.

4.5.2. Performance of the cluster kernel

Now we present the performance of the clustering based kernel (Section 3.2). By only using word cluster information through the cluster kernel, we have achieved a *f*-value of 73.45 with 92.24% precision and 61.02% recall (see Table 3). The precision of the cluster kernel based classifier is much higher compared to the baseline and the class association kernel.

When the suffixes and prefixes and previous tag information are added in the cluster based kernel, the accuracy improves further. Like in class association kernel, here also a word window of length five performs better.

4.5.3. Performance of the composite kernel

As we have seen that the cluster kernel has high precision and the class association kernel has higher recall, we hypothesize that combining these two kernels we can get the advantage of both. The composite kernel is the combination of the class association kernel and the cluster based kernel. These kernels are combined in a weighted fashion using two weight factors: ($K_{composite} = \alpha K_{class\_association} + \beta K_{cluster}$). Selection of the appropriate values of α and β plays an important role in the performance of the composite kernel. During the composite kernel evaluation we have experimented with several values of α and β and selected the best values.

The composite kernel based classifier achieves a *f*-value of 78.99 using only word features (see Table 3). When the affixes and previous tag features are added, the composite kernel achieves a *f*-value of 83.56. When the baseline features are merged (merge strategy), the system yields a *f*-value of 84.62. This is the highest accuracy we have achieved in the SVM based Hindi NER system.

4.6. Comparison of performance

Now we compare the performance of the proposed kernel based SVM classifier with other well-known approaches like MaxEnt, CRF and substring similarity based string kernel. For comparison all these classifiers are trained using same training data and feature set. Table 4 summarizes the results of comparison. In this table we have shown the accuracies obtained using two different feature sets: (a) current and surrounding words and (b) words, suffix, prefix and NE tag of the previous word. During the comparison we have considered the kernel accuracy obtained using only the replacement strategy, when the binary features are not merged with the kernel.

For the comparison we have considered the string sub-sequence kernel proposed by Lodhi et al. (2002). We have used our own implementation of the string sub-sequence kernel where after obtaining the distances between the individual feature values of two particular instances, we merge the distances in a weighted fashion to get the final kernel value between the instances. During the computation of the string kernel also we have considered relative feature importance and relative feature weight factors similar to the proposed kernel. When only word features are used in the string kernel, we achieve a *f*-value of 66.93. We observe that in our Hindi NER task the string kernel suffers from poor precision. We have also prepared the classifiers using the MaxEnt and CRF using similar feature sets. From Table 4 we can observe that when only the word features are used, the proposed composite kernel performs much better than the binary feature based linear SVM, string kernel, MaxEnt and CRF classifiers. When more features are used, then also the kernel outperforms the other classifiers considered in the comparison.

Table 4
Performance comparison for Hindi NER.

Classifier	Feature					
	Words			Words, suf, pref, ptag		
	Pre	Rec	Fval	Pre	Rec	Fval
MaxEnt classifier	66.24	52.57	58.62	85.78	68.14	75.95
CRF classifier	84.02	62.84	71.9	89.47	75.41	81.84
SVM binary feature	81.17	54.04	64.88	86.56	75.36	80.57
SVM string kernel	73.87	61.19	66.93	83.9	74.46	78.89
SVM composite kernel	83.11	75.26	**78.99**	88.41	79.21	**83.56**

5. Experimental results: Biomedical NER

The proposed kernel is also tested in the biomedical NER task. This section presents our experiments on the biomedical NER task. We like to mention here that we have chosen the biomedical NER task in our study only to show that the methodology is very general and is expected to work well in all domains.

For the task we have used the JNLPBA 2004 data (Kim et al., 2004). This corpus is extracted from the GENIA corpus Version 3.02. The training set consists of 2000 abstracts (about 500K words) and the test set contains 404 abstracts (about 100K words). In this data five NE classes are considered: DNA, RNA, protein (Pro), cell-type (CT) and cell-line (CL). The corpus is annotated using BIO format.

For the biomedical NER task we have chosen a set of features that are easy to derive and require no deep knowledge. Most of the features are general features and not specific to the biomedical domain. For the task we have selected a feature set similar to the feature set used by Saha et al. (2009). The features we have used are, word features (current and surrounding words), NE tags of the previous words, a few features that use capitalization and digit information, special characters, word normalization, prefix and suffix information, and parts-of-speech information.

We first used the binary feature based approach to develop the baseline classifier. Then we have used the class association, cluster based and composite kernels and compared the results with the baseline classifier. We have also compared the proposed kernel with the aforementioned string sub-sequence kernel and a feature reduction based MaxEnt classifier (Saha et al., 2009).

Table 5 presents the results when only the word features are used. From the table it is observed that in the biomedical NER task too, the proposed kernel (*f*-value of 61.49) outperforms the binary feature based linear SVM classifier (*f*-value of 57.86). When the string sub-sequence kernel is used in the task (using the implementation used during the Hindi NER experiments), with only word features we achieve a *f*-value of 61.04. Now we have trained a MaxEnt classifier along with the word selection and word clus-

Table 5
The performance of biomedical NER system using only word features.

System	Pro	DNA	RNA	CT	CL	Total
SVM binary features	60.45	50.52	53.28	59.41	38.33	57.86
SVM CA kernel	62.7	53.68	57.94	59.08	37.24	59.54
SVM cluster kernel	62.02	56.17	55.36	65.21	41.66	60.89
SVM composite kernel	62.73	56.66	57.14	65.49	42.11	**61.49**
SVM string kernel	63.63	53.8	57.53	62.53	39.86	61.04
MaxEnt with feature reduction (Saha et al., 2009)	61.07	51.86	51.78	60.83	42.16	58.63

Table 6
The performance of biomedical NER system with complete feature set.

System	Pre	Rec	Fval
Linear SVM with binary features	66.52	66.02	66.27
SVM with the proposed kernel	68.12	67.66	**67.89**
SVM with the string subsequence kernel	67.39	67.65	67.52
MaxEnt based system (Saha et al., 2009)	67.86	66.94	67.41

tering based feature reduction approaches proposed by Saha et al. (2009). Using only the word features, it achieves a *f*-value of 58.63. Hence in our experiments the proposed kernel based SVM classifier performs significantly better than both the MaxEnt classifier and the sub-string similarity based string kernel.

In the biomedical NER task, the string kernel performs better than both the class association and cluster kernels, whereas in the Hindi NER task the individual kernels worked better than the string kernel. This may mean that the substring similarity based string kernel is more appropriate in the biomedical NER task than the Hindi NER task. Another observation is that the class association kernel is not as good in biomedical NER as in Hindi NER. In the biomedical domain, many embedded entities are there where presence of a clue word at the surrounding position of a NE of a particular class modifies it to another class. So the entities are more ambiguous and the vectors are not good enough to represent a class.

Now we have studied the performance of the proposed kernel when a larger feature set, containing all the features considered in this study, is used. Table 6 presents the performance of the system with the complete feature set and related comparisons. Using all the features in binary feature representation, a linear SVM classifier achieves a *f*-value of 66.27. When the proposed composite kernel is used the *f*-value is increased to 67.89. A SVM classifier with the string subsequence kernel achieves a *f*-value of 67.52 in our experiments. As we have used a similar feature set, we have also compared the performance of the system with the MaxEnt based system developed by Saha et al. (2009). Their system achieved the highest *f*-value of 67.41 when the feature dimensionality reduction techniques were applied on the feature set. Hence, although the linear SVM classifier performs poorer than their system, the SVM classifier with the proposed kernel outperforms the system developed by Saha et al. (2009).

6. Conclusion

A novel family of kernel functions is proposed here. The kernel functions compute the weighted distance between a pair of instances. The distance between the string features is computed by making use of certain distance functions, which are obtained from the statistics computed from the entire corpus.

We have experimented with two different types of distance functions. The first one defines a feature vector corresponding to each feature value based on its occurrence in the proximity of named entities. The second distance function makes use of a clustering algorithm. The results look quite promising in both the Hindi and biomedical NER tasks. It is expected that such methods would also work well with other related tasks like parts-of-speech tagging. We will also like to compare this approach to other string based kernels used in literature in the future.

References

Biemann, C., 2006. Chinese whispers - an efficient graph clustering algorithm and its application to natural language processing problems. In: Proc. HLT-NAACL-06 Workshop on Textgraphs.

Borthwick, A., 1999. A maximum entropy approach to named entity recognition. Ph.D. thesis, Computer Science Department, New York University.

Brown, P.F., Pietra, V.J.D., deSouza, P.V., Lai, J.C., Mercer, R.L., 1992. Class-based n-gram models of natural language. Computational Linguistics 18 (4), 467–479.

Collier, N., Nobata, C., Tsujii J., 2000. Extracting the names of genes and gene products with a hidden markov model. In: Proc. COLING 2000, pp. 201–207.

Finkel, J., Dingare, S., Nguyen, H., Nissim, M., Manning C., 2004. Exploiting context for biomedical entity recognition: from syntax to the web. In: Proc. JNLPBA 2004.

Isozaki, H., Kazawa, H., 2002. Efficient support vector classifiers for named entity recognition. In: Proc. COLING-2002, pp. 390–396.

Kazama, J., Makino, T., Ohta, Y., Tsujii, J., 2002. Tuning support vector machines for biomedical named entity recognition. In: Proc. Workshop on NLP in the Biomedical Domain at ACL 2002, pp. 1–8.

Kim, J., Ohta, T., Tsuruoka, Y., Tateisi, Y., Collier, N., 2004. Introduction to the bio-entity recognition task at JNLPBA. In: Proc. JNLPBA 2004, pp. 70–75.

Kim, S., Yoon, J., 2007. Experimental study on a two phase method for biomedical named entity recognition. ICICE Tran. on Information and System E90-D (7), 1103–1110.

Kudo, T., Matsumoto, Y., 2000. Use of support vector learning for chunk identification. In: Proc. CoNLL-2000 and LLL-2000, pp. 142–144.

Kudo, T., Matsumoto, Y., 2001. Chunking with support vector machines. In: Proc. of NAACL-2001, pp. 192–199.

Leaman, R., Gonzalez, G., 2008. Banner: an executable survey of advances in biomedical named entity recognition. In: Proc. Pacific Symposium on Biocomputing, 652–663.

Lee, K.J., Hwang, Y.S., Kim, S., Rim, H.C., 2004. Biomedical named entity recognition using two-phase model based on SVMs. J. Biomedical Infomatics 37 (6), 436–447.

Leslie, C., Eskin, E., Noble W., 2002. The spectrum kernel: a string kernel for SVM protein classification. In: Proc. of Pacific Symposium on Biocomputing.

Leslie, C., Eskin, E., Cohen, A., Weston, J., Noble, W., 2004. Mismatch string kernels for discriminative protein classification. J. Bioinformatics 20 (4), 467–476.

Leslie, C., Kuang, R., 2004. Fast string kernels using inexact matching for protein sequences. J. Machine Learning Research 5 (2004), 1435–1455.

Li, W., McCallum, A., 2004. Rapid development of Hindi named entity recognition using conditional random fields and feature induction. ACM Tran. on Asian Language Information Processing (TALIP) 2 (3), 290–294.

Lin, Y.F., Tsai, T.H., Chou, W.C., Wu, K.P., Sung, T.Y., Hsu, W.L., 2004. A maximum entropy approach to biomedical named entity recognition. In: Proc. 4th Workshop on Data Mining in Bioinformatics, pp. 56–61.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C., 2002. Text classification using string kernels. J. Machine Learning Research 2 (2002), 419–444.

Miller, S., Guinness, J., Zamanian, A., 2004. Name tagging with word clusters and discriminative training. In: Proc. of HLT-04.

Pereira, F., Tishby, N., Lee, L., 1993. Distributional clustering of English words. In: Proc. ACL-1993, pp. 183–190.

Ponomareva, N., Pla, F., Molina, A., Rosso, P., 2007. Biomedical named entity recognition: a poor knowledge HMM-based approach. LNCS 4592, pp. 382–387.

Saha, S.K., Sarkar, S., Mitra, P., 2008. A hybrid feature set based maximum entropy Hindi named entity recognition. In: Proc. IJCNLP-08, pp. 343–349.

Saha, S.K., Sarkar, S., Mitra, P., 2009. Feature selection techniques for maximum entropy based biomedical named entity recognition. J. Biomedical Informatics 42 (5), 905–911.

Settles, B., 2004. Biomedical named entity recognition using conditional random fields and rich feature sets. In: Proc. JNLPBA-2004.

Shen, D., Zhang, J., Zhou, G.D., Su, J., Tan, C.L., 2003. Effective adaptation of a HMM-based named entity recognizer for biomedical domain. In: Proc. ACL 2003 Workshop on NLP in Biomedicine, pp. 49–56.

Singh, A.K., 2008. Named entity recognition for south and south east Asian languages: taking stock. In Proc. IJCNLP-08 Workshop on NER for South and South East Asian Languages, pp. 5–16.

Song, Y., Kim, E., Lee, G.G., Yi, B.K., 2004. POSBIOTM-NER in the Shared Task of BioNLP/NLPBA 2004. In: Proc. JNLPBA-2004.

Takeuchi, K., Collier, N., 2002. Use of support vector machines in extended named entity. In: Proc. CoNLL-2002.

Teo, C.H., Vishwanathan, S., 2006. Fast and space efficient string kernels using suffix arrays. In: Proc. 23rd Int. Conf. on Machine Learning.

Tian, S., Mu, S., Yin, C., 2007. Length-weighted string kernels for sequence data classification. Pattern Recognition Letters 28 (13), 1651–1656.

Tsai, T., Chou, W.C., Wu, S.H., Sung, T.Y., Hsiang, J., Hsu, W.L., 2006. Integrating linguistic knowledge into a CRF framework to identify biomedical named entities. Expert Systems with Applications 30 (1), 117–128.

Ushioda, A., 1996. Hierarchical clustering of words. In: Proc. COLING 96, pp. 1159–1162.

Vanschoenwinkel, B., Liu, F., Manderick, B., 2005. Weighted kernel functions for SVM learning in string domains: a distance function viewpoint. In: Proc. Int. Conf. on Machine Learning and Cybernetics.

Vapnik, V.N., 1995. The Nature of Statistical Learning Theory. Springer-Verlag, New York.

Zhou, G.D., Su, J., 2004. Exploring deep knowledge resources in biomedical name recognition. In: Proc. JNLPBA-2004, pp. 96–99.