International Journal of Innovative Computing, Information and Control
Volume **16**, Number **2**, April **2020**

ICIC International ©2020 ISSN 1349-4198
pp. **555–570**

TOWARDS WORD SENSE DISAMBIGUATION USING MULTIPLE KERNEL SUPPORT VECTOR MACHINE

Liyun Zhong and Tinghua Wang

School of Mathematics and Computer Science
Gannan Normal University
Economic & Technological Development Zone, Ganzhou 341000, P. R. China
wth2003nc@163.com

Received August 2019; revised December 2019

Abstract. *Word sense disambiguation (WSD), the task of identifying the intended meanings (senses) of words in context, has been a long-standing research objective for natural language processing (NLP). In this paper, we investigate the problem of combining multiple feature channels using kernel methods for the purpose of effective WSD. A straightforward method is to use a uniform combination of more adequate kernels, which are built from different types of data representations or knowledge sources. Instead of using an equal weight for all kernels in the combination, we consider the problem of integrating multiple feature channels using the state-of-the-art multiple kernel learning (MKL) approach, which can learn different weights that reflect the different importance of the feature channels for disambiguation. This approach has the advantage of the possibility to combine and select the more relevant feature channels in an elegant way. Combined with the support vector machine (SVM), this approach is demonstrated with several Senseval/Semeval disambiguation tasks.*

Keywords: Word sense disambiguation (WSD), Multiple kernel learning (MKL), Support vector machine (SVM), Kernel method, Natural language processing (NLP)

1. **Introduction.**

Ambiguity is inherent to human language. Particularly, word sense ambiguity is prevalent in all natural languages, with a large number of words having more than one meaning. For instance, the English noun bank can mean "sloping raised land, especially along the sides of a river" or "an organization where people and businesses can invest or borrow money, convert to foreign money, etc. or a building where these services are offered". The correct sense of an ambiguous word can be determined based on the context where it occurs, and correspondingly the problem of word sense disambiguation (WSD) is defined as the task of automatically assigning the most appropriate meaning to a polysemous word in a given context [1]. As a fundamental semantic understanding task at the lexical level in natural language processing (NLP), WSD can benefit many applications such as machine translation, information retrieval, parsing, and question answering. Machine translation uses the concept of WSD to make correct lexical choices and information retrieval uses it to solve the ambiguity in the queries. In actual applications, WSD is often fully integrated into the system and cannot be separated out (for instance, in information retrieval, WSD is often not done explicitly but is just a by-product of query to document matching). WSD is considered to be a key step in order to approach language understanding beyond keyword matching [2]. Although WSD for human is essentially a subconscious process and presents no difficulties, it is very difficult

DOI: 10.24507/ijicic.16.02.555

to formalize the computational process of disambiguation since it is classified among "AI-complete" problems [3], that is, it is a task whose solution is at least as hard as the most difficult problems in artificial intelligence.

WSD methods can be generally classified into two types: knowledge-based and machine learning [1,4]. Knowledge-based WSD systems exploit the information in a lexical knowledge base, such as WordNet and Wikipedia, to perform WSD. These approaches usually pick the sense whose definition is most similar to the context of the ambiguous word, by means of textual overlap or using graph-based measures [2,5,6]. Machine learning approaches, also called corpus-based approaches, do not make use of any knowledge resources for disambiguation. These approaches range from supervised learning in which a classifier is trained for each distinct word on a corpus of manually sense-annotated examples, to completely unsupervised methods that cluster occurrence of words, thereby inducing senses. Recent advances in WSD have benefited greatly from the availability of corpora annotated with word senses. Most accurate WSD systems to date exploit supervised methods which automatically learn cues useful for disambiguation from manually sense-annotated data.

For machine learning-based WSD systems, commonly used algorithms include Naïve Bayesian model, decision trees, maximum entropy, support vector machine (SVM), and so on. Among them, kernel methods in general and SVM [7,8] in particular have demonstrated excellent performance in terms of accuracy and robustness and applying kernel methods to the WSD task is a very promising choice [9-14]. The approach used by kernel methods is to map the input data into a feature space by means of kernel function and then uses learning algorithm to discover relations in the space. Specifically, kernel methods work by mapping the data from the input space into a high-dimensional (possibly infinite) feature space, which is usually chosen to be a reproducing kernel Hilbert space (RKHS), and then building linear algorithms in the feature space to implement nonlinear counterparts in the input space. The mapping, rather than being given in an explicit form, is determined implicitly by specifying a kernel function, which computes the inner product between each pair of data points in the feature space. There are several reasons that make kernel methods applicable to WSD and other NLP problems [13]. Firstly, instead of manual construction of feature space for the learning task, kernel functions provide an alternative way to design useful features in the feature space automatically, therefore, ensuring necessary representational power. Secondly, kernel methods offer a flexible and efficient way to define application-specific kernels for introducing background knowledge and modeling explicitly linguistic insights. This property allows to notably improve the performance of the general learning methods and their simple adaptation to the specific application. Lastly, kernel methods can be naturally applied to the non-vectorial types of data, thus taking into account the structure of the data and greatly reducing the need for careful feature engineering in these structures.

From the point of view of modularization, kernel methods consist of two main components, namely the kernel and actual learning algorithm. The kernel can be considered as an interface between the input data and the learning algorithm, and is the key component to ensure the good performance of kernel methods [7,15]. Actually, for real applications, kernel is the only task-specific component of kernel methods. In the domain of WSD, the widely used kernel is the "Bag of Words" (BOW) kernel [7,13], which is based on the BOW representation of the context in which an ambiguous word occurs. In this representation, each word or term constitutes a dimension in a vector space, independent of other terms in the same context. Despite its ease of use, this kernel suffers from well-known limitations, mostly due to its inability to exploit semantic similarity between terms: contexts sharing terms that are different but semantically related will be considered as unrelated.

To address this problem, a number of attempts have been made to incorporate semantic knowledge into the BOW kernel, resulting in the so-called semantic kernels [7,8,10-13]. Besides the BOW representation of the context in which an ambiguous word occurs, more structured representations, such as strings, trees and graphs, can be also employed to represent word contexts [5,6,9,13]. Lodhi et al. [16] proposed the string kernel, which is significantly different from the BOW model, for effective text categorization which can be seen as the general domain of WSD. In string kernel, features are not terms, but all possible ordered subsequences of characters occurring in documents. The string kernel was later extended to the word-sequences kernel [17,18], which is used to compare the sequences of words. Compared with the string kernel, the word-sequences kernel greatly reduces the average length of symbols per document, which yields a significant improvement in computational efficiency. Moreover, matching word sequences allows working with more linguistically meaningful symbols. For WSD, Giuliano et al. [9] modified the generic definition of the sequence kernel to enable it to recognize the collocations in a local window of the word to be disambiguated. Specifically, they defined two new kernels: the N-gram collocation kernel and N-gram part-of-speech (POS) kernel, which is defined as a sequence kernel applied to sequences of lemmata and POSs around the word to be disambiguated, respectively. The third kind of kernel used for text categorization is the tree kernel [19,20], which is a powerful way to encode the syntactic structure information of the input documents in the form of parse trees. The main underlying idea of tree kernels is to count the number of tree substructures common to both parse trees.

Generally, the BOW kernel, sequence kernel and tree kernel capture the word frequency, syntagmatic relation and syntactic structure information, respectively. The features that are extracted from the context in which an ambiguous word occurs have different characteristics and can be considered as heterogeneous, which motivates the use of multiple kernel learning (MKL) [21-24] for classification problems. Giuliano et al. [9] first proposed using kernel combination approach for WSD. The benefit of kernel combination is that it allows to integrate heterogeneous sources of information in a simple and effective way. Although it has been shown in [9] that it is quite possible to substantially improve the disambiguation performance using a combination of more adequate kernels (each kernel in the combination being adequate to the source of information represented by the part of the features that the kernel uses), it used an equal weight for each kernel in the combination and thus ignored the fact that different kernels associated with different input data representations make different contributions to the WSD task. Instead, we can use weights that reflect the importance of knowledge source that the kernel uses for disambiguation via MKL. In addition, since all the presented WSD systems do not exploit the syntactic information produced by a parser, we can integrate such information by adding a tree kernel in the framework MKL. To summarize, in this paper, we apply the newly proposed MKL approach [24] to integrate the multiple sources of information provided by different representations for robust WSD. MKL aims to learn an optimal combination of a set of predefined base kernels in order to identify a good target kernel for the applications. Since the base kernels can be built from different types of data representations, the MKL approach has the advantages of the possibility to combine and select the most relevant data representation in an elegant way.

The rest of this article is outlined as follows. Section 2 briefly introduces MKL in the SVM framework. Section 3 introduces our proposed approach which includes a detailed description of the individual kernels and how to learn the convex combination of these basic kernels. Experimental results are reported in Section 4, followed by some concluding remarks in Section 5.

2. Preliminaries.

2.1. Support vector machine.
Ambiguity is inherent to human language. Particularly, word sense ambiguity is prevalent in an SVM which is a theoretically well motivated algorithm developed from statistical learning theory and has shown impressive performance in many fields [7,8]. Suppose we are given a set of labeled training samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{l}$ in a binary classification problem, where $\boldsymbol{x}_i \in \mathrm{X} \subset \mathrm{R}^n$ (R denotes the set of real numbers) is the input data and $y_i \in \{+1, -1\}$ is the corresponding class label. The goal of the SVM is to find an optimal hyperplane $\boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}) + b = 0$ that separates the training points into two classes with the maximal margin, where $\boldsymbol{w}$ is the normal vector of the hyperplane, b is a bias, and ϕ is a feature map which maps $\boldsymbol{x}_i$ to a high-dimensional feature space. This hyperplane can be obtained by solving the following optimization problem

$$\begin{aligned} \min\ & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{l}\xi_i \\ \text{s.t.}\ & y_i\left(\boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}_i) + b\right) \geq 1 - \xi_i \\ & \xi_i \geq 0,\ i = 1, \ldots, l \end{aligned} \tag{1}$$

where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_l)^{\mathrm{T}}$ is the vector of slack variables and C is the regularization parameter used to impose a trade-off between the training error and generalization. In order to solve the SVM optimization problem, suppose α_i be the Lagrange multiplier corresponding to the i-th inequality in (1), the dual problem of (1) is shown to be

$$\begin{aligned} \max\ & \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{s.t.}\ & \sum_{i=1}^{l}\alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq C,\ i = 1, \ldots, l \end{aligned} \tag{2}$$

where $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j)$ is the kernel function which implicitly defines the feature map ϕ. After the solution is obtained, the SVM decision function is given by

$$f(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l}\alpha_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b\right) \tag{3}$$

where the samples $\boldsymbol{x}_i$ with $\alpha_i > 0$ are called support vectors.

For multiclass classification problems, there are several approaches available to extend binary SVM to multiclass SVM [11,12,25]. These approaches roughly fall into two categories. The first denoted as all-in-one or single machine is to directly consider all data in one optimization formulation. The second involves considering a decomposition of a multiclass problem into several binary subproblems and then combining their solutions. There are two widely used strategies to decompose a multiclass problem: one-versus-rest (1-v-r) and one-versus-one (1-v-1). Given a problem with d classes, the 1-v-r strategy constructs d binary SVMs, in which each of them is trained to separate one class from the other classes, while the 1-v-1 strategy constructs $d(d-1)/2$ binary SVMs, in which each of them is trained to separate one class from another class. When a test sample is provided, it is applied to all the binary SVMs and their outputs are combined based on some voting techniques, such as "MaxWins" voting scheme which counts how often each class is output by the binary SVMs and the test sample is then assigned to the most voted class. It has been shown that when the parameters of SVM are properly tuned and selected both approaches usually present no significant difference in terms of classification accuracy. However, the decomposition approach is often recommended for practical use because of conceptual simplicity and lower computational overhead.

The last two decades have witnessed an explosion of the use of SVM for WSD [9,11-14,26-30]. Among them, we highlight the following work. Cabezas et al. [26] presented a supervised word sense tagger using SVM. Their system was designed for performing WSD independent of the language of lexical samples provided for Senseval-2 task. Lee et al. [28] used SVM learning and multiple knowledge sources to perform WSD for Sensenval-3 English lexical sample task and multilingual lexical sample task. Pahikkala et al. [29] explored the capability of SVM for the gene versus protein name disambiguation task. Giuliano et al. [9] presented a semi-supervised technique for WSD using SVM that exploited external knowledge acquired in an unsupervised manner. In addition, regarding the disambiguation performance, SVM has been shown to achieve the best results compared with several supervised approaches [13,27,30].

2.2. **Multiple kernel learning.** Instead of formulating an optimization criterion with a fixed kernel k, one can leave the kernel k as a combination of a set of predefined kernels, which results in the problem of MKL [21-23]. MKL maps each sample to a multiple-kernel-induced feature space and a linear classifier is learned in this space. The feature mapping used in MKL takes the form of $\phi(\cdot) = [\phi_1^{\mathrm{T}}(\cdot), \ldots, \phi_M^{\mathrm{T}}(\cdot)]^{\mathrm{T}}$, which is induced by M predefined base kernels $\{k_m(\cdot,\cdot)\}_{m=1}^{M}$ with alternative kernel forms or kernel parameters. The linear combination of these kernels is given by

$$k = \sum_{m=1}^{M} \mu_m k_m \tag{4}$$

where μ_m is the corresponding combination coefficient. Let $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_M)^{\mathrm{T}} \in \Delta$, where Δ is the domain of $\boldsymbol{\mu}$. By varying the constraint on $\boldsymbol{\mu}$, different MKL models can be obtained. For example, when $\boldsymbol{\mu} \in \Delta$ lies in a simplex, i.e.,

$$\Delta = \left\{ \boldsymbol{\mu} : \|\boldsymbol{\mu}\|_1 = \sum_{m=1}^{M} \mu_m = 1, \mu_m \geq 0 \right\} \tag{5}$$

we call the L_1-norm of kernel weights, and the resulting model is the L_1-MKL. Most MKL methods fall into this category. When

$$\Delta = \left\{ \boldsymbol{\mu} : \|\boldsymbol{\mu}\|_p \leq 1, p > 1, \mu_m \geq 0 \right\} \tag{6}$$

we call the L_p-norm of kernel weights, and the resulting model is L_p-MKL. A special case is the L_2-norm of kernel weights and the resulting model L_2-MKL.

The idea of MKL can be generally applied to many kinds of kernel methods, such as the commonly used SVM and kernel fisher discriminant analysis (KFDA), leading to SVM-based MKL and discriminant MKL, respectively. Our work in this paper will mainly focus on the SVM-based MKL formulations. Like SVM, the dual problem of SVM-based MKL can be represented as

$$\begin{aligned} \max\ & \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j \sum_{m=1}^{M} \mu_m k_m(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{s.t.}\ & \sum_{i=1}^{l} \alpha_i y_i = 0,\ \boldsymbol{\mu} \in \Delta, \\ & 0 \leq \alpha_i \leq C,\ i = 1, \ldots, l \end{aligned} \tag{7}$$

The goal of training MKL is to learn μ_m, α_i and b with the given M base kernels, and the final decision function is given by

$$f(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i y_i \sum_{m=1}^{M} \mu_m k_m(\boldsymbol{x}_i, \boldsymbol{x}) + b \right) \tag{8}$$

Compared with traditional kernel methods employing a fixed kernel, MKL demonstrates flexibility in automated kernel learning and also reflects the fact that typical learning problems often involve multiple, heterogeneous data sources. In the following, we aim at using MKL for robust WSD, which does not only consider the uniform combination of different kernels.

3. **Word Sense Disambiguation Using Multiple Kernel Learning.**

In this section we use MKL to combine heterogeneous sources of information that we found relevant for WSD. For each of these aspects it is possible to define kernels independently.

3.1. **Bag-of-words kernel.**

In the machine learning-based WSD systems, the features extracted from the contexts are usually in the bag-of-words (BOW) representation which reduces a text to a histogram of word frequencies. Formally, let t_0 denote the word to be disambiguated and $\boldsymbol{x} = (t_{-bl}, \ldots, t_{-1}, t_1, \ldots, t_{br})$ be the context of t_0, where $t_{-bl}, \ldots, t_{-1}$ are the words in the order they appear preceding t_0, and correspondingly $t_1, \ldots, t_{br}$ are the words that follow t_0 in the context. Consider that we are also given a vocabulary V consisting of n words, which can be extracted from all the contexts in the training corpus. The BOW model (also called vector space model, VSM) [7] of the context $\boldsymbol{x}$ is defined as follows:

$$\phi : \boldsymbol{x} \rightarrow \phi(\boldsymbol{x}) = (tf(t_1, \boldsymbol{x}), \ldots, tf(t_n, \boldsymbol{x}))^{\mathrm{T}} \in \mathrm{R}^n \tag{9}$$

where $tf(t_i, \boldsymbol{x})$, $1 \leq i \leq n$, is the frequency of the occurrence of the word t_i in the context $\boldsymbol{x}$. If we consider the feature space defined by the VSM, the BOW kernel is given by the inner product between the feature vectors:

$$k_{\mathrm{BOW}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_j) = \sum_{t \in V} tf(t, \boldsymbol{x}_i) tf(t, \boldsymbol{x}_j) \tag{10}$$

Note that in this case the input space is the set of contexts.

BOW model is probably one of the simplest constructions used in text processing. Despite its ease of use, the BOW model suffers from the well-known limitation: although the contexts are represented by the words they contain, the word positions, orders and other grammatical information in the context are lost. In addition, semantic information disappears as well, hence contexts sharing words that are different but semantically related are considered as unrelated. To overcome such limitations, many researchers have explored the extensions or improvements of the BOW kernel including word position-sensitive kernels [31], semantic kernels [7,9,11-14] and so on.

3.2. **Sequence kernel.**

Sequences (or strings) of symbols naturally occur in text analysis and other areas such as bioinformatics. As far as text analysis is concerned, very often we have to deal with the sequences: words are sequences of characters, syntagmatic relations are established by sequences of words. In general, the strategy of modeling syntagmatic relation is to extract bigrams and trigrams of the collocated words as features to describe the local contexts. Focusing on WSD, most of the methods utilize a very rich feature set, including bigrams and trigrams in the local context of the word to be disambiguated [1]. However, syntagmatic features are very difficult to be modeled, because they typically generate huge feature spaces when extracted explicitly. In addition, non-consecutive or shifted collocations are very hard to represent.

As an alternative, sequence kernels (or string kernels) [7,16-18] can be used to analyze the syntagmatic relations. Rather than making use of features such as word frequencies, sequence kernels use the number of all possible ordered subsequences contained in a text. In general, sequence kernels count the number of (possibly non-contiguous) matching subsequences shared by two sequences. Non-contiguous occurrences are penalized according to the number of gaps they contain. Formally, let Σ be a finite alphabet and $x = s_1 s_2 \cdots s_{|x|}$ be a sequence over Σ (i.e., $s_i \in \Sigma$, $1 \le i \le |x|$). Let $\boldsymbol{i} = [i_1, i_2, \ldots, i_n]$, with $1 \le i_1 \le i_2 \le \cdots \le i_n \le |x|$, be a subset of the indices in x, we denote the subsequence $s_{i_1} s_{i_2} \cdots s_{i_n}$ by $x[\boldsymbol{i}] \in \Sigma$. Note that $x[\boldsymbol{i}]$ does not necessarily form a contiguous subsequence of x. The length spanned by $x[\boldsymbol{i}]$ in x is $len(\boldsymbol{i}) = i_n - i_1 + 1$. The sequence kernel of order n for the pair of sequences x and z over Σ is defined as follows:

$$k_n(x, z) = \sum_{u \in \Sigma^n} \sum_{\boldsymbol{i}: x[\boldsymbol{i}] = u} \sum_{\boldsymbol{j}: z[\boldsymbol{j}] = u} \lambda_1^{len(\boldsymbol{i}) + len(\boldsymbol{j})} \tag{11}$$

where $\lambda_1 \in (0, 1]$ is a decay factor used to penalize non-contiguous subsequences. When $\lambda_1 = 1$ there is no penalization of gaps, meaning every subsequence will contribute equally to the kernel value whether the elements in the sequence are contiguous or not. As λ_1 decreases the gap penalization increases, meaning that when $\lambda_1 \to 0$ the kernel will be reduced to counting the number of consecutive subsequences. This definition is the so-called gap-weighted subsequence kernel, which was initially introduced to compare sequences of characters [16] and was later extended to compare sequences of words [17,18].

The basic idea behind the sequence kernel is that two sequences are similar if they have in common many subsequences. For WSD, we restrict the generic definition of the sequence kernel to recognize the collocations in a local context of the word to be disambiguated. In particular, the new kernel is defined as an up-to-n gap-weighted subsequence kernel applied to the sequences of lemmata around the word to be disambiguated:

$$k_{\text{seq}}(x, z) = \sum_{i=1}^{n} k_i(x, z) \tag{12}$$

This formulation allows us to estimate the number of common subsequences of lemmata (i.e., collocations) between two samples (contexts), in order to capture the syntagmatic similarity. This kernel depends on the parameters n (the length of the non-contiguous subsequences) and λ_1 (the decay factor). For example, when $n = 2$, this kernel allows us to represent all (sparse) bigrams in the local context of a specified word.

The limitation of such sequence kernel is very conspicuous, i.e., only exact lemma-matches contribute to the similarity. To solve this problem, two alternative soft-matching criteria based on WordNet synonymy and domain proximity were used to improve the kernel [9]. Both criteria are based on the assumption that every word in a sentence can be substituted by another preserving the original meaning, if these words are paradigmatically related (e.g., synonyms or domain related words). For example, if we consider the terms “Ronaldo” and “football star” as equivalent, then the sentence “Ronaldo scores the first goal” is equivalent to “The football star scores the first goal”, providing a strong evidence to disambiguate the verb “score” in the first sentence.

As for the computational complexity, an explicit computation of (11) is unfeasible even for small values of n. Fortunately, it can be efficiently calculated by a dynamic programming algorithm with a complexity of $o(n\,|x|\,|z|)$ [16]. It is worth noting that when computing the order-n kernel $k_n(x, z)$, this algorithm computes all kernels $k_i(x, z)$ for $i < n$ as intermediaries, which offers the possibility to compute (12) at almost no extra cost.

3.3. **Tree kernel.**

A tree is defined as a connected directed graph with no cycles. We denote trees as $T_1, T_2, \ldots$, tree nodes as $n_1, n_2, \ldots$, and the set of nodes in the tree T_i as N_{T_i}. Let $F = \left\{f_1, f_2, \ldots, f_{|F|}\right\}$ be the set of tree fragments (substructures) and $I_i(n)$ an indicator function which is equal to 1 if the target f_i is rooted at node n and equal to 0 otherwise. The main idea of tree kernels [19,20] is to compute the number of common substructures between two trees T_1 and T_2 without explicitly considering the whole fragment space. Formally, the tree kernel over T_1 and T_2 is given by

$$k_{\text{tree}}(T_1,T_2) = \sum_{n_1\in N_{T_1}}\sum_{n_2\in N_{T_2}}\Psi(n_1,n_2) = \sum_{n_1\in N_{T_1}}\sum_{n_2\in N_{T_2}}\sum_{i=1}^{|F|} I_i(n_1)I_i(n_2) \tag{13}$$

where the Ψ function is equal to the number of common fragments rooted in nodes n_1 and n_2, and thus, depends on the fragment type. We report its algorithm for evaluation of the number of syntactic tree fragments since the structures we here work with are parse trees. In a parse tree, each node with its children is associated with the execution of a grammar production rule. A syntactic tree fragment is a set of nodes and edges from the original tree which is still a tree and with the constraint that any node must have all or none of its children. The algorithm for efficiently computing Ψ is as follows:

Step 1. If the productions at n_1 and n_2 are different, then $\Psi(n_1,n_2)=0$;

Step 2. If the productions at n_1 and n_2 are same, and n_1 and n_2 have only leaf children (i.e., they are pre-terminal symbols), then $\Psi(n_1,n_2)=\lambda_2$;

Step 3. If the productions at n_1 and n_2 are same, and n_1 and n_2 are not pre-terminals, then $\Psi(n_1,n_2)=\lambda_2\prod_{i=1}^{nc(n_1)}\left(1+\Psi\left(ch_{n_1}^i,ch_{n_2}^i\right)\right)$.

$nc(n_1)$ is the number of children of n_1, ch_n^i is the i-th child of node n and λ_2 is a decay factor penalizing larger structures. Note that, since the productions are same, $nc(n_1)=nc(n_2)$.

3.4. **The proposed MKL approach.** WSD can be viewed as a problem of classifying each word, according to its surrounding context, to one of its senses. Therefore it is a classification problem with a few classes, i.e., multiclass classification problem. We here apply MKL [21-23] approach to WSD. Given the base kernels discussed above, a composite kernel can be created as follows:

$$\begin{aligned} k(\boldsymbol{x},\boldsymbol{z}) \;=\; & \mu_1\frac{k_{\text{BOW}}(\boldsymbol{x},\boldsymbol{z})}{\sqrt{k_{\text{BOW}}(\boldsymbol{x},\boldsymbol{x})k_{\text{BOW}}(\boldsymbol{z},\boldsymbol{z})}} + \mu_2\frac{k_{\text{seq}}(\boldsymbol{x},\boldsymbol{z})}{\sqrt{k_{\text{seq}}(\boldsymbol{x},\boldsymbol{x})k_{\text{seq}}(\boldsymbol{z},\boldsymbol{z})}} \\ & + \mu_3\frac{k_{\text{tree}}(\boldsymbol{x},\boldsymbol{z})}{\sqrt{k_{\text{tree}}(\boldsymbol{x},\boldsymbol{x})k_{\text{tree}}(\boldsymbol{z},\boldsymbol{z})}} \end{aligned} \tag{14}$$

where $\mu_1+\mu_2+\mu_3=1$, $\mu_i\geq 0$ $(i=1,2,3)$. This model integrates essentially the multiple sources of information provided by different context representations. In some sense, the parameters μ_i (kernel weights) impose a tradeoff among the word frequency, syntagmatic relation and syntactic structure information of contexts for WSD. Moreover, since kernel weights reflect the importance of the different knowledge sources for WSD, this composite kernel allows us to introduce *a priori* knowledge in the WSD systems by designing specific μ_i and to extract some useful information from the best tuned μ_i parameters. In addition, due to the different scales of the values of the three individual kernels, they are normalized before combination. This can avoid one kernel value being overwhelmed by that of another one. For example, for the sequence kernel, since the number of subsequences associated with a sequence increases with its length, the value of the kernel is highly dependent on the size of the sequences to be compared. In order to compensate for this size effect, it is commonplace to consider the normalized version of the kernel.

MKL approaches can be generally classified into two types: one-stage MKL and two-stage MKL [21-23]. The one-stage MKL algorithms learn both the optimal weights for kernel combination and the SVM solution by solving a joint optimization problem while two-stage MKL algorithms first learn the optimal kernel weights according to certain criteria, then applies the learned optimal kernel to train a kernel classifier. Compared with one-stage MKL algorithms, two-stage MKL algorithms generally achieve comparable or even better classification performance, while incurring much less computational cost. In our recently published work [24], we presented an effective two-stage MKL algorithm based on the notion of Hilbert-Schmidt independence criterion (HSIC) Lasso [32]. In discussing the connection between MKL and HSIC Lasso, we found that the proposed algorithm not only has a clear statistical interpretation that minimum redundant kernels with maximum dependence on output labels are found and combined, but also can efficiently compute the global optimal solution by solving a Lasso optimization problem. We here apply this two-stage MKL approach to determine the kernel weights, i.e., parameters μ_i, and train the SVM classifier using the learned composite kernel for robust WSD.

Mathematically, let $\bar{\mathbf{L}} = \mathbf{HLH}$ and $\bar{\mathbf{K}} = \mathbf{HKH}$, where $\mathbf{K}$, $\mathbf{L}$ and $\mathbf{H}$ are the kernel matrix for input data which is defined as $\mathbf{K}_{ij} = k(x_i, x_j)$, kernel matrix for output labels which is defined as $\mathbf{L} = \boldsymbol{y}\boldsymbol{y}^{\mathrm{T}}$ ($\boldsymbol{y} = (y_1, \dots, y_l)^{\mathrm{T}}$), and centering matrix which is defined as $\mathbf{H} = \mathbf{I} - \mathbf{e}\mathbf{e}^{\mathrm{T}}/l$ ($\mathbf{I}$ is the identity matrix and $\mathbf{e} = (1, \dots, 1)^{\mathrm{T}}$), respectively. The combination coefficient $\boldsymbol{\mu}$ can be estimated by the HSIC Lasso:

$$\begin{aligned} \min \ & \frac{1}{2}\left\| \bar{\mathbf{L}} - \sum_{m=1}^{M} \mu_m \bar{\mathbf{K}}_m \right\|_{\mathrm{F}}^2 + \lambda_3 \|\boldsymbol{\mu}\|_1 \\ \text{s.t.} \ & \mu_1, \dots, \mu_M \geq 0 \end{aligned} \tag{15}$$

where $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm and $\lambda_3 > 0$ is the regularization parameter. We sketch the overall WSD procedure using MKL in Algorithm 1, where the centered kernel matrix can be calculated by

$$\bar{\mathbf{K}}_{ij} = \mathbf{K}_{ij} - \frac{1}{l}\sum_{i=1}^{l} \mathbf{K}_{ij} - \frac{1}{l}\sum_{j=1}^{l} \mathbf{K}_{ij} + \frac{1}{l^2}\sum_{i=1}^{l}\sum_{j=1}^{l} \mathbf{K}_{ij} \tag{16}$$

Readers can refer to Reference [24] for detailed information.

```
Algorithm 1. WSD using MKL
Input: Data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{l}$, base kernels
$\{k_m(\cdot,\cdot)\}_{m=1}^{3} = \left\{ \frac{k_{\mathrm{BOW}}(\boldsymbol{x},\boldsymbol{z})}{\sqrt{k_{\mathrm{BOW}}(\boldsymbol{x},\boldsymbol{x})k_{\mathrm{BOW}}(\boldsymbol{z},\boldsymbol{z})}}, \frac{k_{\mathrm{seq}}(\boldsymbol{x},\boldsymbol{z})}{\sqrt{k_{\mathrm{seq}}(\boldsymbol{x},\boldsymbol{x})k_{\mathrm{seq}}(\boldsymbol{z},\boldsymbol{z})}}, \frac{k_{\mathrm{tree}}(\boldsymbol{x},\boldsymbol{z})}{\sqrt{k_{\mathrm{tree}}(\boldsymbol{x},\boldsymbol{x})k_{\mathrm{tree}}(\boldsymbol{z},\boldsymbol{z})}} \right\}$,
and regularization parameters $C$ and $\lambda_3$.
Output: SVM classifier $f(\boldsymbol{x})$.
1: Initialize $\boldsymbol{\mu} = \mathbf{e}/3$.
2: Calculate the kernel matrix $\mathbf{L} = \boldsymbol{y}\boldsymbol{y}^{\mathrm{T}}$.
3: Calculate the centered kernel matrices $\bar{\mathbf{L}}$ and $\{\bar{\mathbf{K}}_m\}_{m=1}^{3}$.
4: Obtain $\boldsymbol{\mu}$ by solving (15).
5: Normalize each element of $\boldsymbol{\mu}$ as $\mu_m \leftarrow \mu_m / \sum_{m=1}^{3} \mu_m$.
6: Combine the kernel matrices using the weight $\boldsymbol{\mu}$ and train an SVM classifier.
```

4. **Experimental Evaluation.**

This experiment evaluates the performance of the proposed method with several Senseval/Semeval[1] benchmark examples. Senseval/Semeval is an international organization devoted to the evaluation of WSD systems. We consider four kernels, i.e., BOW kernel, sequence kernel, tree kernel, and the proposed composite

[1]http://www.senseval.org/

kernel for comparison. These kernels are embedded in the SVM classifier individually. Besides, we use the MFS (most frequent sense) method as the baseline model, which selects the most frequent sense in the training data as the answer, independently of the contexts of the ambiguous word.

4.1. **Experimental setup.** We select the data sets for four words, namely interest, line, hard and serve, which have been used in numerous comparative studies of WSD. Tables 1-4 show the detailed descriptions and distributions of different senses of these four words. The interest data [33] consists of 2368 instances where the noun interest is used in one of six senses taken from the Longman Dictionary of Contemporary English (LDOCE). The instances are extracted from the part of speech tagged subset of the Penn Treebank Wall Street Journal Corpus (ACL/DCI version). The line data [34] consists of 4147 instances where the noun line is used in one of six possible WordNet senses. This data was extracted from the 1987-1989 Wall Street Journal (WSJ) corpus and the American Printing House for the Blind (APHB) corpus. The distribution of senses is somewhat skewed with more than 50% of the instances used in the product sense while all the other instances more or less equally distributed among the other five senses. The hard data [35] consists of 4333 instances taken from the San Jose Mercury News Corpus (SJM) and are annotated with one of three senses of the adjective hard, from WordNet. The distribution of instances is skewed with almost 80% of the instances used in the not easy-difficult sense. The serve data [35] consists of 4378 instances with the verb serve as the target word. They are annotated with one of four senses from WordNet. Like line data it was created from the WSJ and APHB corpora.

Table 1. Description and distribution of senses of *interest*

Sense	Sense tag frequency	Percentage (%)
readiness to give attention	361	15.24
quality of causing attention to be given to	11	0.46
activity, etc. that one gives attention to	66	2.79
advantage, advancement or favor	178	7.52
a share in a company or business	500	21.11
money paid for the use of money	1252	52.87

Table 2. Description and distribution of senses of *line*

Sense	Sense tag frequency	Percentage (%)
cord	373	8.99
division	374	9.02
formation	349	8.42
phone	429	10.34
product	2218	53.48
text	404	9.74

Table 3. Description and distribution of senses of *hard*

Sense	Sense tag frequency	Percentage (%)
not easy (difficult)	3455	79.74
not soft (metaphoric)	502	11.59
not soft (physical)	376	8.68

Table 4. Description and distribution of senses of *serve*

Sense	Sense tag frequency	Percentage (%)
supply with food	1814	41.43
hold an office	1272	29.05
function as something	853	19.48
provide a service	439	10.03

For each data set, we partition it into a training set and a test set: 70% of the data set is used for training and the rest for prediction. Stratified sampling is used to preserve the ratio of different classes in the train and test files. The data set was preprocessed using the text mining infrastructure (TMI) [36]. The preprocessing includes sentence boundary determination, stop word removal and stemming. We used the stemmer and stop word list embedded in the TMI. After the proper preprocessing, we used the LIBSVM package[2] to train and test the SVM model. We adopted the following parameterization of the considered kernels:

- There is no parameter to tune for the BOW kernel. It is evident that the BOW kernel is essentially a linear kernel. In fact, in WSD and text categorization applications, the number of features of a data set is usually large, even much larger than the number of instances. In this case, one may not need to map data to a higher dimensional space since the nonlinear mapping does not improve the performance [37].
- The gap penalization factor λ_1 and length n of the subsequences in sequence kernel are taken from $\{0.1, 0.5, 1.0\}$ and $\{2, 3\}$, respectively. Recall from Section 3.2 that the smaller the value of λ_1, the bigger the gap penalization. As a result, when $\lambda_1 \to 0$, gap-weighted subsequence kernel reduces to kernel based on contiguous subsequences. In practice, we do not observe significant variations when λ_1 decreases below 0.1.
- The decay factor λ_2 for tree kernel is taken from $\{0.001, 0.01, 0.1, 0.5\}$. Since a parse tree is an ordered tree, each document, that is a sequence of sentences, is represented as an ordered tree of ordered trees. Thus a document can be a tree where each root's child is the parse tree of a sentence and the leaves are its word's lemma. Nevertheless, we do not use the complete parse tree, but only the nodes of the following word types: noun, proper noun, adjective, verb, pronoun and adverb.
- The regularization parameter λ_3 for the proposed MKL is taken from $\{10^{-2}, 10^{-1}, \dots, 10^{2}\}$.

In addition, the parameters of all kernels as well as the SVM parameter C, which is taken from $\{10^{-2}, 10^{-1}, \dots, 10^{2}\}$, are optimized by 5-fold cross-validation on the training set. For example, for the proposed MKL approach, we perform 5-fold cross-validation by grid-search over two dimensions, i.e., $C = \{10^{-2}, 10^{0}, \dots, 10^{2}\}$ and $\lambda_3 = \{10^{-2}, 10^{-1}, \dots, 10^{2}\}$.

4.2. **Results and discussion.** We measure the classification performance using the F_1 score, which is a popular measure for comparing performances of different algorithms typically on the data with skewed class distribution. We recall that, given a confusion matrix for a class c such as the one in Table 5, the precision of a classifier, for the class c, is the ratio of the number of texts correctly assigned to c to the total number texts assigned to c. The recall is the ratio of the number of texts correctly assigned to c to the total number of texts belonging to c. The F_1 score is the harmonic mean of the precision

[2]http://www.csie.ntu.edu.tw/∼cjlin/libsvm

and the recall. Formally, these measures are shown as follows:

$$\text{Precision}(c) = \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c} \tag{17}$$

$$\text{Recall}(c) = \frac{\text{TP}_c}{\text{TP}_c + \text{FN}_c} \tag{18}$$

$$F_1(c) = \frac{2 * \text{Precision}(c) * \text{Recall}(c)}{\text{Precision}(c) + \text{Recall}(c)} \tag{19}$$

For multiclass classification problems, the micro-average and the macro-average are used to evaluate the classifier. Table 6 gives the formulae for the precision and recall. According to the kind of average used, the global F_1 score is then given by

$$F_1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{20}$$

Table 5. Confusion matrix for class c

	Positive	Negative
Predicted positive	TP_c	FP_c
Predicted negative	FN_c	TN_c

Table 6. Micro-average and macro-average for d classes

	Micro	Macro
Precision	$\frac{\sum_c \text{TP}_c}{\sum_c (\text{TP}_c + \text{FP}_c)}$	$\frac{\sum_c \text{precision}(c)}{d}$
Recall	$\frac{\sum_c \text{TP}_c}{\sum_c (\text{TP}_c + \text{FN}_c)}$	$\frac{\sum_c \text{recall}(c)}{d}$

The average classification results in terms of the micro- and macro-F_1 over 10 trials are summarized in Tables 7 and 8, respectively. Figures 1 and 2 provide more straightforward illustration of performance comparison using F_1 measures with different methods. The bold numbers in Tables 7 and 8 denote the best performance of these methods on each data set. To conduct a rigorous comparison, the paired t-test [38] is performed. The paired t-test is used to analyze whether the difference between two compared algorithms on one data set is significant. The p-value of the paired t-test represents the probability that two sets of compared results come from distributions with an equal mean. A p-value of 0.05 is considered to be statistically significant. The win-tie-loss (W-T-L) summarizations based on t-test are attached at the bottoms of Tables 7 and 8, where the proposed WSD using MKL (denoted by WSD-MKL) and other methods are respectively compared. In comparing two algorithms such as algorithm I versus algorithm II, a win or loss means that algorithm I is better or worse than algorithm II on a data set. A tie means that both algorithms achieve the same performance.

From these tables and figures, we find that all kernels produce significantly better classification performances than the MFS baseline. More importantly, for all data sets we see that the proposed WSD-MKL achieves significant performance improvement over the other three kernels. Take the *interest* data set for example: WSD-MKL achieves the micro-averaged F_1 value of 87.83% whereas the BOW kernel, sequence kernel, and tree kernel achieve that of 86.00%, 85.19%, and 84.23%, respectively. WSD-MKL achieves

Table 7. Micro-averaged F_1 values for the considered five methods

Data set	Micro-F_1 (%)				
	MFS	BOW kernel	Sequence kernel	Tree kernel	WSD-MKL
interest	52.87	86.00	85.19	84.23	**87.83**
line	53.48	82.94	80.93	79.70	**84.72**
hard	79.74	83.70	83.86	82.54	**85.81**
serve	41.43	86.29	85.74	84.63	**88.96**
W-T-L	4-0-0	4-0-0	4-0-0	4-0-0	–

Table 8. Macro-averaged F_1 values for the considered five methods

Data set	Macro-F_1 (%)				
	MFS	BOW kernel	Sequence kernel	Tree kernel	WSD-MKL
interest	11.53	64.21	62.32	61.73	**72.63**
line	11.62	75.21	74.86	73.10	**78.44**
hard	14.87	33.13	33.75	31.48	**35.86**
serve	9.76	55.21	54.93	53.64	**60.47**
W-T-L	4-0-0	4-0-0	4-0-0	4-0-0	–

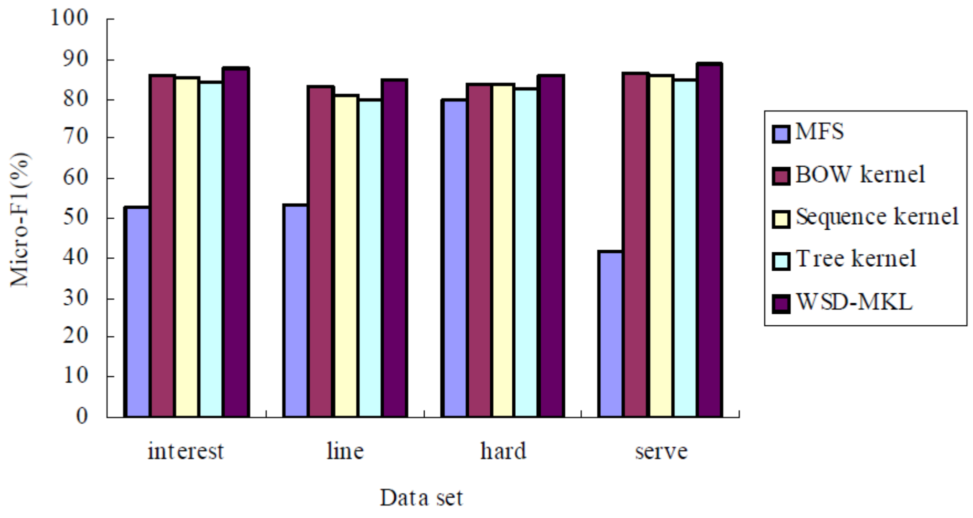

Figure 1. Comparison of micro-averaged F_1 using different methods

the macro-averaged F_1 value of 72.63% with relative improvements of 13.11% ((72.63 − 64.21)/64.21), 16.54% ((72.63 − 62.32)/62.32), and 17.66% ((72.63 − 61.73)/61.73) over the BOW kernel, sequence kernel, and tree kernel, respectively. This indicates that the BOW representation, sequence representation and syntactic-tree representation of the context of the word to be disambiguation are complementary and the proposed WSD-MKL approach can well integrate them. Furthermore, we find that the tree kernel is consistently outperformed by other two kernels (BOW kernel and sequence kernel). This implies that, compared with the word frequency and syntagmatic relation information, the syntactic structure is not competitive for WSD.

It is also worth noting that, due to the severely skewed class distribution of the data sets, for all methods the micro-averaged F_1 values are consistently higher than the macro-averaged F_1 values. Conceptually, the micro-averaged F_1 will not be affected by the small

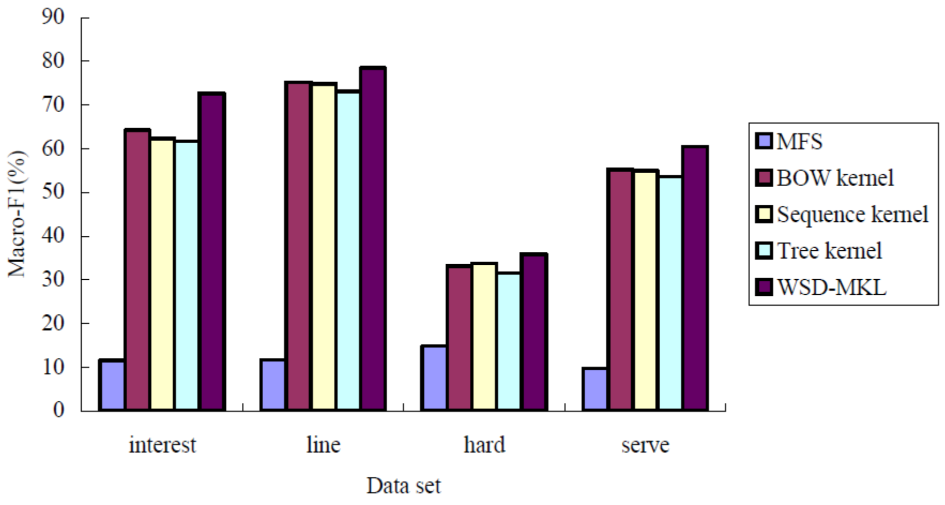

FIGURE 2. Comparison of macro-averaged F_1 using different methods

classes since it gives an equal weight to all instances. On the contrary, the macro-averaged F_1 is an average over all the classes so the small classes will drastically affect the value.

In summary, compared with the single kernel SVM, the proposed multiple kernel SVM can significantly improve the performance of WSD tasks. We attribute this to the fact that, as a typical learning problem, WSD often involves multiple, heterogeneous data sources or representations and MKL can automatically combine and learn the most relevant representations for disambiguation. Furthermore, for every application of SVM, users have to choose which kernel to use and sometimes even have to design their own kernels. Is it possible to alleviate choosing kernels or designing specialized kernels? It is evident that MKL is a great step towards that solution.

5. **Conclusions.** We have presented a WSD method based on the MKL framework, which first combines multiple and heterogeneous sources of information in a global kernel function and then trains an SVM classifier using the obtained global kernel. This MKL approach can well explore and combine the word frequency, syntagmatic relation and syntactic structure information, and therefore outperforms the kernels that take into account the word frequency, syntagmatic relation or syntactic structure information individually. To our knowledge, no other work has so far combined the multiple properties of context in such a principled way for WSD. Possible extensions of this work include the theoretical verification of the superior performance of the proposed approach, conducting more experiments on the lexical-sample and all-words tasks, as well as extending to other semantic analysis tasks to further verify the effectiveness of the proposed approach.

Acknowledgment. This work is partially supported by the National Natural Science Foundation of China (No. 61966002) and the Natural Science Foundation of Jiangxi Province of China (No. 20192BAB207016). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] R. Navigli, Word sense disambiguation: A survey, *ACM Computing Surveys*, vol.41, no.2, pp.1-69, 2009.

[2] E. Agirre, O. L. Lacalle and A. Soroa, Random walks for knowledge-based word sense disambiguation, *Computational Linguistics*, vol.40, no.1, pp.57-84, 2014.

[3] D. Y. Turdakov, Word sense disambiguation methods, *Programming and Computer Software*, vol.36, no.6, pp.309-326, 2010.

[4] B. Krawczyk and B. T. McInnes, Local ensemble learning from imbalanced and noisy data for word sense disambiguation, *Pattern Recognition*, vol.78, pp.103-119, 2018.

[5] R. Navigli and M. Lapata, An experimental study of graph connectivity for unsupervised word sense disambiguation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.32, no.4, pp.678-692, 2010.

[6] E. A. Corrêa Jr, A. A. Lopes and D. R. Amancio, Word sense disambiguation: A complex network approach, *Information Sciences*, vols.442-443, pp.103-113, 2018.

[7] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, New York, USA, 2004.

[8] Z. Wang, Q. Zeng and M. Lv, Modeling the external truck arrivals in container terminals based on DBN and SVM, *ICIC Express Letters*, vol.12, no.10, pp.1033-1040, 2018.

[9] C. Giuliano, A. Gliozzo and C. Strapparava, Kernel methods for minimally supervised WSD, *Computational Linguistics*, vol.35, no.4, pp.513-528, 2009.

[10] T. Pahikkala, S. Pyysalo, J. Boberg, J. Järvinen and T. Salakoski, Matrix representations, linear transformations, and kernels for disambiguation in natural language, *Machine Learning*, vol.74, no.2, pp.133-158, 2009.

[11] T. Wang, J. Rao and Q. Hu, Supervised word sense disambiguation using semantic diffusion kernel, *Engineering Applications of Artificial Intelligence*, vol.27, pp.167-174, 2014.

[12] T. Wang, W. Li, F. Liu and J. Hua, Sprinkled semantic diffusion kernel for word sense disambiguation, *Engineering Applications of Artificial Intelligence*, vol.64, pp.43-51, 2017.

[13] X. Li, S. Qing, H. Zhang, T. Wang and H. Yang, Kernel methods for word sense disambiguation, *Artificial Intelligence Review*, vol.46, no.1, pp.41-58, 2016.

[14] W. Zhu, Semi-supervised word sense disambiguation using von Neumann kernel, *International Journal of Innovative Computing, Information and Control*, vol.13, no.2, pp.695-701, 2017.

[15] T. Wang and W. Li, Kernel learning and optimization with Hilbert-Schmidt independence criterion, *International Journal of Machine Learning and Cybernetics*, vol.9, no.10, pp.1707-1717, 2018.

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

[17] N. Cancedda, E. Gaussier, C. Goutte and J.-M. Renders, Word-sequences kernels, *Journal of Machine Learning Research*, vol.3, pp.1059-1082, 2003.

[18] N. Cancedda and P. Mahé, Factored sequence kernels, *Neurocomputing*, vol.72, nos.7-9, pp.1407-1413, 2009.

[19] S. Bloehdorn and A. Moschitti, Combined syntactic and semantic kernels for text classification, *Proc. of the 29th European Conference on Information Retrieval*, Rome, Italy, pp.307-318, 2007.

[20] T. Goncalves and P. Quaresma, Text classification using tree kernels and linguistic information, *Proc. of the 7th International Conference on Machine Learning and Applications*, San Diego, USA, pp.763-768, 2008.

[21] M. Gönen and E. Alpayın, Multiple kernel learning algorithms, *Journal of Machine Learning Research*, vol.12, pp.2211-2268, 2011.

[22] S. S. Bucak, R. Jin and A. K. Jain, Multiple kernel learning for visual object recognition: A review, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.36, no.7, pp.1354-1369, 2014.

[23] Y. Gu, J. Chanussot, X. Jia and J. A. Benediktsson, Multiple kernel learning for hyperspectral image classification: A review, *IEEE Transactions on Geoscience and Remote Sensing*, vol.55, no.11, pp.6547-6565, 2017.

[24] T. Wang and X. Tu, Multiple kernel learning using nonlinear Lasso, *IEEJ Transactions on Electrical and Electronic Engineering*, vol.14, no.5, pp.760-767, 2019.

[25] C. W. Hsu and C. J. Lin, A comparison of methods for multiclass support vector machines, *IEEE Transactions on Neural Networks*, vol.13, no.2, pp.415-425, 2002.

[26] C. Cabezas, P. Resnik and J. Stevens, Supervised sense tagging using support vector machines, *Proc. of the 2nd International Workshop on Evaluating Word Sense Disambiguation Systems*, Toulouse, France, pp.59-62, 2001.

[27] Y. K. Lee and H. T. Ng, An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation, *Proc. of the Conference on Empirical Methods in Natural Language Processing*, Philadelphia, USA, pp.41-48, 2002.

[28] Y. K. Lee, H. T. Ng and T. K. Chia, Supervised word sense disambiguation with support vector machines and multiple knowledge sources, *Proc. of the 3rd International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, Spain, pp.137-140, 2004.

[29] T. Pahikkala, F. Ginter, J. Boberg, J. Järvinen and T. Salakoski, Contextual weighting for support vector machines in literature mining: An application to gene versus protein name disambiguation, *BMC Bioinformatics*, vol.6, no.1, pp.157-168, 2005.

[30] Z. Zhong and H. T. Ng, It makes sense: A wide-coverage word sense disambiguation system for free text, *Proc. of the ACL System Demonstrations*, Uppsala, Sweden, pp.78-83, 2010.

[31] T. Pahikkala, S. Pyysalo, F. Ginter, J. Boberg, J. Järvinen and T. Salakoski, Kernels incorporating word positional information in natural language disambiguation tasks, *Proc. of the 18th International Florida Artificial Intelligence Research Society Conference*, Menlo Park, USA, pp.442-447, 2005.

[32] M. Yamada, W. Jitkrittum, L. Sigal, E. P. Xing and M. Sugiyama, High-dimensional feature selection by feature-wise kernelized Lasso, *Neural Computation*, vol.26, no.1, pp.185-207, 2014.

[33] R. F. Bruce and J. Wiebe, Word-sense disambiguation using decomposable models, *Proc. of the 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, USA, pp.139-146, 1994.

[34] C. Leacock, G. Towell and E. Voorhees, Corpus-based statistical sense resolution, *Proc. of the ARPA Workshop on Human Language Technology*, Plainsboro, USA, pp.260-265, 1993.

[35] C. Leacock, G. A. Miller and M. Chodorow, Using corpus statistics and WordNet relations for sense identification, *Computational Linguistics*, vol.24, no.1, pp.147-165, 1998.

[36] L. E. Holzman, T. A. Fisher, L. M. Galitsky, A. Kontostathis and W. M. Pottenger, A software infrastructure for research in textual data mining, *International Journal on Artificial Intelligence Tools*, vol.13, no.4, pp.829-849, 2004.

[37] C. W. Hsu, C. C. Chang and C. J. Lin, A practical guide to support vector classification, *Technical Report*, Department of Computer Science, National Taiwan University, 2003.

[38] J. Demšar, Statistical comparisons of classifiers over multiple data sets, *Journal of Machine Learning Research*, vol.7, pp.1-30, 2006.