



Word sense disambiguation using semantic kernels with class-based term values

Berna ALTINEL^{1,*}, Murat Can GANİZ¹, Bilge ŞİPAL², Erencan ERKAYA¹

Onur Can YÜCEDAĞ¹, Muhammed Ali DOĞAN¹

¹Computer Engineering Department, Faculty of Engineering, Marmara University, İstanbul, Turkey

²Department Mathematics and Computer Science, Faculty of Science and Letters, İstanbul Kültür University, İstanbul, Turkey

Received: 20.05.2018

Accepted/Published Online: 26.03.2019

Final Version: 26.07.2019

Abstract: In this study, we propose several semantic kernels for word sense disambiguation (WSD). Our approaches adapt the intuition that class-based term values help in resolving ambiguity of polysemous words in WSD. We evaluate our proposed approaches with experiments, utilizing various sizes of training sets of disambiguated corpora (SensEval¹). With these experiments we try to answer the following questions: 1.) Do our semantic kernel formulations yield higher classification performance than traditional linear kernel?, 2.) Under which conditions a kernel design performs better than others?, 3.) Does the addition of class labels into standard term-document matrix improve the classification accuracy?, 4.) Is their combination superior to either type?, 5.) Is ensemble of these kernels perform better than the baseline?, 6.) What is the effect of training set size? Our experiments demonstrate that our kernel-based WSD algorithms can outperform baseline in terms of F-score.

Key words: Word sense disambiguation, semantic kernel, classification, term relevance values, sprinkling

1. Introduction

Polysemy is the capacity of a word to have multiple meanings. It is like a one-to-many association among objects of database models; since one single word might have more than one meaning which could probably be different depending on the context it appears. WSD is the task of automatically finding the appropriate sense of a word depending on its context. It is expected that the falsely disambiguated words would jeopardize the performance of text classification by including noise; on the other hand, correctly disambiguated words could improve the performance of text classification. For instance, Table 1 lists five different sentences. It is easy to understand that sentences with id_1 , id_3 and id_4 are about the subject under the category of computer and sentences id_2 and id_5 are about the subject under the category of animals. According to the traditional bag of words (BOW), which is a well-known feature representation technique that only regards the frequency of the words, a basic similarity calculation such as Cosine or Jaccard among sentences id_1 , id_3 , and id_4 will be zero; since they have no words in common. The same situation is valid for the similarity between sentences id_2 and id_5 . On the other hand, the similarity between sentences id_1 and id_2 will probably be greater than zero since they shared a word; “mouse”. Moreover, although they convey different messages, the similarity between sentences id_2 and id_4 will probably be greater than zero since they shared a word “cell”.

*Correspondence: berna.altinel@marmara.edu.tr

¹<http://www.senseval.org/>

Table 1. Sentences with polysemous words

Id	Sentence
1	A mouse is used to move a cursor on a computer screen
2	Experiments revealed the same cells that have also been discovered in the eyes of rat, mouse and hamster.
3	Laptops have several different capabilities including the touchpad, the trackball and the pointing stick which pupils enjoy.
4	Two common battery types, the 6-cell and the 9-cell, are manufactured for this notebook.
5	The pupil, a hole located in the center of the iris of the eye that allows light to strike the retina, has been discovered to have similar structure in many animals.

There is an extensive bibliography for resolving ambiguity of polysemous words. Besides, of many different kinds of approaches, several types of semantic kernels are used in WSD classifiers. These kernels can be grouped into five: latent kernels [1, 2], domain kernels [3, 4], sequence kernels [2, 5], knowledge-based kernels [6–9], and composite kernels [10, 11].

Latent kernels can be defined as kernels to expose and use latent values among words [1, 2]. Domain kernels attempt to incorporate domain information of words into the disambiguation process [3, 4]. Sequence kernels or string kernels count the number of contiguous subsequences shared by two sequences while penalizing the number of noncontiguous subsequences shared by them and then compute a similarity value for those two sequences [5, 12]. Knowledge-based kernels take advantages of a treasure, dictionary or a knowledge source such as WordNet [13], Wikipedia² for WSD [6–8]. In addition, composite kernels are actually weighted summation kernels of individual two valid kernels [10, 11]. The existence of theoretical or experimental studies on text classification, including further information extracted from corpus-based statistics or a lexical dictionary like WordNet, or a treasure like Wikipedia is expected to increase the classification performance of textual materials. This could probably be caused by the usage of such resources or corpus-based statistics, which gives opportunity to expose hidden relationships among words and documents.

Semantic kernels have potential to expose hidden relationships among words and documents. Class weighting kernel (CWK) [14] and relevance values kernel (RVK) [14] are previously proposed semantic kernels applied to text classification. In this paper, we firstly evaluate these two semantic kernels on our WSD system. After that, we enriched these two semantic kernels with sprinkled features (i.e. class labels of the documents in the training corpus) by directly adding each label as a binary valued column into term-document matrix and use this augmented matrix in calculating our semantic matrix. Another algorithm we evaluate on our WSD system is a composite kernel, which is actually a weighted combination of CWK and RVK. The main contributions of this work are:

- We applied CWK [14] and RVK [14] kernels, which are originally developed for text classification, into WSD in finding the correct sense of polysemous words according to their context. In addition, to the best

² <http://www.wikipedia.org/>

of our knowledge, this is the first attempt to use class-based term weighting and relevance values to build semantic kernels for WSD.

- We adapt sprinkled features into semantic kernels that use class-based term weighting and relevance values for WSD, which is again the first attempt as far as we know.
- We present a novel composite kernel by combining CWK and RVK. We perform several experiments in order to find the best combination parameters.
- In addition, we build an ensemble system of all of our suggested semantic kernels whose individual decisions are combined to improve the performance of the overall system.

The remainder of the paper is organized as follows: The following section provides a background to WSD, term weighting techniques, semantic kernels, sprinkling and ensemble techniques. An overview of the related works is given in Section 3. Then, Section 4 presents the proposed methodologies in detail. Experimental setup, results and discussions on these are given in Section 5. The final section presents future work and conclusions drawn from this study.

2. Background

2.1. Word sense disambiguation

Word sense disambiguation is about determining the correct sense of a polysemous word (i.e. words with more than one distinct meaning) according to the context in which it occurs. For instance, the noun key could be defined as "metal device shaped in such a way that when it is inserted into the appropriate lock, the lock's mechanism can be rotated" for the context it occurs: "I've lost my car keys." or alternatively "something crucial for explaining" in the context of "Hard work is the key to success." or "list of answers to a test" in the context it occurs "some students had stolen the key to the final exam" (WordNet). WordNet lists 14 distinct senses for the noun key. WSD actually is a classification problem in natural language processing since it attempts to predict the most suitable sense of a word given a context among its possible senses defined by the dictionary. Consequently, the objective of a WSD task is to identify the most suitable sense s_i for the word t_w for its context from its all-possible senses-set as shown in (1):

$$sense_t_w = \{s_1, s_2, s_3, \dots, s_n\}: s_i \in sense_t_w \quad (1)$$

2.2. Term weighting methods

Term Weighting with Abstract Features: A novel term weighting technique is proposed in [16]. According to Biricik et al. [16], the influence of a word in a class is computed as shown in Eq. (2):

$$W_{w,c} = \log(tfc_{w,c} + 1) * \log\left(\frac{N}{N_w}\right), \quad (2)$$

where $tfc_{w,c}$ shows the total term frequency of a word w in the documents of class c , N denotes the total number of documents in the corpus, and N_w represents the total number of documents which contain term w . Biricik et al. [16] implement this class-dependent term weighting method and they conduct a series of experiments on benchmark textual datasets. According to the experimental results reported in [16], their methodology with this

class-dependent term weighting technique improves the classification performance compared to other existing methods.

Term Frequency-Relevance Frequency (TF-RF): Lan et al. [17] present term frequency-relative frequency (TF-RF) and it pays attention to only the occurrence number of documents that include this word. According to their study [17], a selected category is labeled as the positive category while all the other categories in the same dataset are altogether labeled as the negative category. In TF-RF the more concentrated a high-frequency word is in the positive category than in the negative category, the more influence it has in selecting the positive instances from the negative instances. The formula of TF-RF is represented in Eq. (3):

$$TF - RF = tf_w * \log(2 + \frac{a}{\max(1, c)}), \quad (3)$$

where tf_w shows the term frequency of word w , a denotes the number of documents in the positive category which include word w , and c is the number of documents in the negative category which include word w . According to an explanatory example given in [17]; in contrast to inverse document frequency, with RF methodology each word is assigned more appropriate weights from the point of different categories since RF takes care of the category information.

2.3. Linear kernel vs semantic kernels

Linear kernel has been widely used in text classification domain since it is the simplest kernel function. As represented in Eq. (4), the calculated kernel value between two documents, namely d_p and d_q , depends on the inner products of feature vectors of the documents as shown in Eq. (4):

$$K(d_p, d_q) = d_p * d_q. \quad (4)$$

Thus, a linear kernel captures similarity between documents based on the words they share. This is a problem since it does not consider semantic relations between terms and documents. This can be addressed by including semantic information between words using semantic kernels. In semantic kernel, documents are enriched with some semantic information originated from an ontology such as WordNet, Wikipedia, Wiktionary [18–23], statistical calculations from higher-order paths [24–26], class-based term weighting in a supervised setting [14], class-based term weighting in a semisupervised setting [15], and exponential transformation in semantic diffusion [27, 28]. Other examples of semantic kernels are latent semantic kernels [12] or domain kernels [3]. For instance, the study in [19] uses super-concept declaration in semantic kernels. Their aim is to create a kernel algorithm which captures the knowledge of topology that belongs to their super-concept expansion. They utilize this mapping with the help of a semantic smoothing matrix S that is composed of P and P^T which contains super-concept information about their corpus. Their suggested kernel function is given in Eq. (5):

$$K(d_p, d_q) = d_p * P * P^T * d_q. \quad (5)$$

Different choices of semantic proximity matrix S leads to different variants of semantic kernels. Semantic kernels reduce disadvantages of traditional text classification and improve the prediction abilities in comparison with standard linear kernels.

2.4. Sprinkling

Sprinkling is a process of adding further terms representing class labels to training documents in order to augment class-based relationships in training phase. For instance in [29], latent semantic indexing (LSI) is performed both on standard term-document matrix and term-document matrix augmented with sprinkled terms. The sprinkling process is shown in Figure 1:

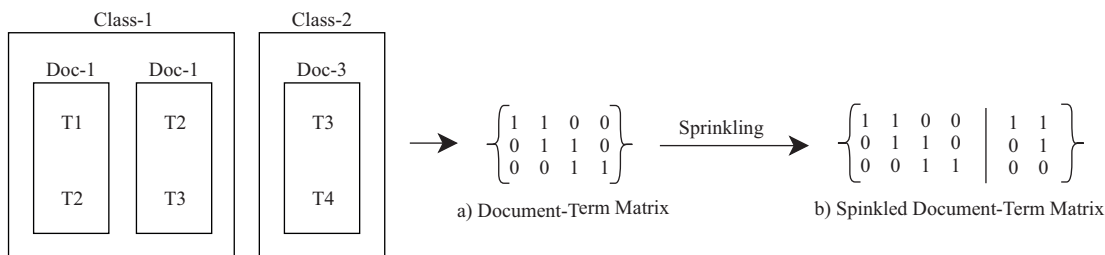


Figure 1. Sprinkling

In Figure 1, to explain the sprinkling process, we use the toy corpus from [28] that has 2 different class labels with 3 documents (Doc-1, Doc-2, and Doc-3) and 4 different terms (t_1, t_2, t_3 , and t_4). In (a) we get the document-term matrix with 3 documents and 4 terms. In (b) we apply the sprinkling and get an augmented document-term matrix. These new additional columns show the class labels of the corresponding documents. For instance, the first and the second documents belong to class 1 which is encoded in the fifth column and the third document belongs to class 2 which is encoded in the sixth column of the augmented matrix in (b).

Chakraborti et al. [29] drop sprinkling terms after performing LSI. Test documents are classified using k-nearest neighbors (kNN) with Euclidean distance metric. Experimental results show that the presence of sprinkling terms increase the classification accuracy. For instance, the classification accuracies of sprinkled LSI are reported as 86.99%, 80.60%, 80.42%, and 93.89% while the classification accuracies of LSI are reported as 79.32%, 72.55%, 66.30%, and 91.17%; respectively on 20 NewsGroups³ dataset in [29]. They state that the integration of further knowledge which represents the latent class structure improves the classification performance. In other words, with the help of sprinkling process documents associated with the same class label are drawn closer to each other. Moreover, Chakraborti et al. [29] have taken the concept of sprinkling into one step further: adaptive sprinkling. In standard sprinkling process, the number of sprinkling terms are the same for all of the pairs of the classes. However, this is not a perfect solution since some classes are more easily separable than others in real-world situations. In adaptive sprinkling process, the number of sprinkling terms should depend on the complexity of the class decision boundary. Chakraborti et al. [29] decided the number of sprinkling terms for each class pairs based on the confusion matrix. Their empirical evaluation results show that adaptive sprinkling improves the classification accuracies of kNN with Cosine distance metric, kNN with Euclidean distance metric, LSI, and support vector machines (SVM) [29].

2.5. Ensemble techniques

An ensemble algorithm combines a set of classifier models whose decisions are combined to increase the performance of the individual classifiers [30]. The motivation for using ensembles is to emphasize the strengths of different classification models while diluting their weaknesses. Experimental studies show that ensemble

³<http://www.cs.cmu.edu/textlearning>

systems are often more accurate than the individual base learner that make them up in classification domain [31–33]. Lately several theoretical explanations have been offered to justify the effectiveness of some commonly used ensemble techniques [34, 35]. Adaptive boosting (AdaBoost) [36], AdaBoost.M1 which is an ensemble generated by many weak classifiers have a weighted error no greater than 0.5 [37], AdaBoost.M2 [37] which is a multiclass extension of AdaBoost.M1, majority voting [38, 39] Bayesian-based decision rules [40], fuzzy aggregation methods [41], and bagging [42].

3. Related work

A wide range of kernel methods has been developed for WSD. They are latent kernels, domain kernels, sequence kernels, knowledge-based kernels, and composite kernels.

Latent Kernels:

Latent semantic kernels can be defined as kernels to expose and use latent values among words [1, 2]. LSA kernels are examples of this type. Very recently, Wang et al. [27] proposed a semantic diffusion kernel, which considers all possible paths connecting two nodes in the graph. According to their experiments, their diffusion kernel is superior to LSI and linear kernel with several SensEval disambiguation tasks such as interest, line, hard, and serve. The authors also expanded their work by adding class labels into their previous semantic diffusion kernel model [28]. Their kernel is named as sprinkled semantic diffusion kernel and the idea is to add the class labels of the documents in the original term-document matrix and then use this augmented term-document in their previous semantic diffusion kernel.

Domain kernels:

It has been shown [4] and mentioned [3] that domain information is important to disambiguate polysemous words. For instance, there is a polysemous word mouse in the following sentences: Use a mouse to move a cursor on a computer screen. Experiments revealed the same cells that have also been discovered in rat, mouse, and hamster. Table 2 represents a simple example of domain values for the words computer, hamster, mouse, screen and rat. The ambiguity of the words mouse can be resolved by considering the domain of the context in which it appears.

Table 2. A simple example of domain values for the words computer, hamster, mouse, screen, and rat.

	Animals	Computer
computer	0	1
hamster	1	0
laptop	0	1
mouse	0.5	0.5
screen	0	1
rat	1	0

According to simple BOW feature representation of these two sentences, the similarity of them is zero because they have no words in common. Use a mouse to move a cursor on a computer screen. The laptop has

been broken. On the other hand, the following two sentences have words in common. Therefore, the similarity of them is greater than zero according to simple BOW feature representation, even though they convey different messages. Experiments revealed the same cells that have also been discovered in rats, mouse, and hamsters. I have lost my mouse pad. Using domain tables, the similarity between "Use a mouse to move a cursor on a computer screen." and "The laptop has been broken." will be greater than zero since the words mouse, laptop, screen, and computer are in the same domain. A domain matrix can be generated from lexical resources such as WordNet [43]. Gliozzo et al. [44] generated domain matrix by singular value decomposition. Then they developed a domain kernel that simply uses the domain values of the words. The domain kernel computes the domain similarities among texts.

Sequence (syntagmatic) kernels: It was analyzed and reported that contiguous structures of sequences are very powerful to resolve the ambiguity of a polysemous word [45, 46]. Yarowsky looked at the adjacent words of a polysemous word and reported that these left and right adjacents are very powerful indicators to resolve the ambiguity of a polysemous word [46]. Sequence kernels or string kernels count the number of contiguous subsequences shared by two sequences. They generally penalize the number of noncontiguous subsequences shared by them and then compute a similarity value for those two sequences [2, 5]. Shawe-Taylor and Cristianini [2] defined two types of sequence kernels for WSD: the n-gram collocation kernel and n-gram part of speech (PoS) kernel. The n-gram collocation kernel is a gap-weighted subsequence kernel which has been applied to sequences in the around of the polysemous word. In a similar way, PoS kernel has been defined as the sequences of PoSs around the polysemous word [2].

Knowledge-based kernels: These kernels take advantages of a dictionary or a knowledge-resource for WSD [6–9]. Scott and Matwin [7] incorporate semantic information from a hierarchical thesaurus in their text classifier and stated that this thesaurus information jeopardize the classification performance on their experimental environment including Reuters-21578 and DigiTrad datasets. Furthermore, Bloehdorn and Hotho [6] use hypernyms from WordNet to enrich their feature space. They achieved satisfactory results on Reuters-215781 and OHSUMED⁴ datasets. Banerjee and Pedersen [9] reported that the effect of using WordNet for WSD on classification performance depends on the size of the training set; if it is small the usage of WordNet improves the classification performance; however, if it is large the usage of WordNet jeopardizes the classification performance. Mavroeidis et al. [47] present an unsupervised WSD approach, which uses an outer hierarchical tree to extract the pathwise distances of the words in a given text. They mentioned that adjacent words in a given text are semantically close to each other and this closeness is directly proportional to their pathwise distance in the hierarchical tree. They conducted several experiments on SensEval-2 and SensEval-3 datasets and reported that their approach has a value for WSD since it increases the classification performance.

Composite Kernels: A composite kernel is actually a weighted summation kernel as in Eq. (6):

$$k_{com}(d_1, d_2) = \lambda k_a(d_1, d_2) + (1 - \lambda)k_b(d_1, d_2) \quad (6)$$

, where λ is a positive real number ($0 < \lambda < 1$), $k_a(d_1, d_2)$ is kernel similarity score between documents d_1 and d_2 according to kernel function k_a , $k_b(d_1, d_2)$ is kernel similarity score between documents d_1 and d_2 according to kernel function k_b and $k_{com}(d_1, d_2)$ is the composite kernel similarity score between documents d_1 and d_2 . $k_{com}(d_1, d_2)$ is again a valid kernel because of the closure properties of kernels [2]. Recent works [10, 11] has empirically shown that composite kernels constantly increases the performance of the individual ones. Wang

⁴ <http://disi.unitn.it/moschitti/corpora.htm>

et al. [27] have presented a composite kernel that combines the BOW kernel and sequence kernel in order to advance the classification performance of SVM in WSD. According to their experimental results on two popular WSD corpora, interest and serve, the composite kernel of BOW kernel and sequence kernel achieves higher classification accuracy than both individual BOW kernel and sequence kernel. Other methods for WSD can be found in recent literature studies such as [3] and [8].

4. Approach

4.1. WSD using semantic kernels

In this study, we first evaluate two previously implemented semantic kernels; CWK [14] and RVK [14] on our WSD system. CWK is inspired by the class-based term weighting calculation in Eq. (2) as described in Section 2.2. Similarly, RVK mainly depends on TF-RF, which is a supervised term weighting method as mentioned in Section 2.2. Inspired by TF-RF, RVK is designed to benefit from the class-based term relevance weights in the semantic smoothing kernel [15]. Both of these class-based term calculations contribute to the classifier to give more emphasis on "core" words of each class, which are strictly associated to the subject of that class while decreasing the importance on the "general" words, which may have similar distributions on different classes. Since these types of weighting matrices have extra information related to the terms compared to BOW representation, these results expose semantic similarities among words and documents by smoothing the representation of the text documents. The general architecture of both CWK and RVK is shown in Figure 2.

To enrich the standard linear kernel function by semantic proximity between terms, we generate the semantic proximity matrix P using class-based term weights in CWK as shown in Eq. (2) and using class-based term relevance weights in RVK as shown in Eq. (3). In order to generate square semantic similarity matrix, we just multiply P matrix with its transpose form, P^T . A kernel function must satisfy Mercer's condition as mentioned in [48]. These conditions are satisfied when the Gram matrix is positive semidefinite. It has been shown in [12] that the matrix G formed by the kernel function in Eq. (7) with the outer matrix product is indeed a positive semidefinite matrix.

$$k_{RVK}(d_1, d_2) = d_1 S S^T d_2^T, \quad (7)$$

In this methodology S is a semantic proximity matrix which is used to convert documents from the input space to a feature space.

4.2. WSD using semantic kernels with sprinkled terms

We also implemented sprinkled CWK, which has nearly the same architecture as CWK and sprinkled RVK that is based on RVK. The main difference between CWK, RVK and sprinkled CWK, sprinkled RVK is that sprinkled CWK and sprinkled RVK use term-document matrix augmented with sprinkled terms as mentioned in Section 2.4 while standard CWK and RVK use original term-document matrix. The architecture of sprinkled CWK and sprinkled RVK is given in Figure 3.

4.3. WSD using composite semantic kernel

We also evaluated our WSD system with composite form of CWK and RVK. Both kernel (gram) matrices generated from CWK and RVK have similarity information among documents. However, since these values are

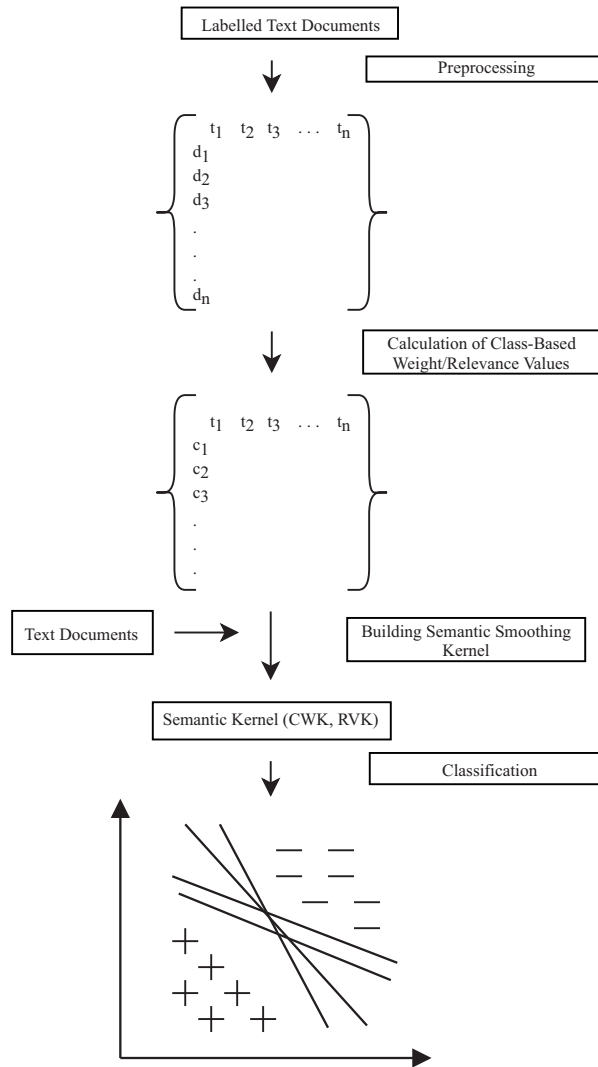


Figure 2. The architecture of semantic kernel (CWK, RVK).

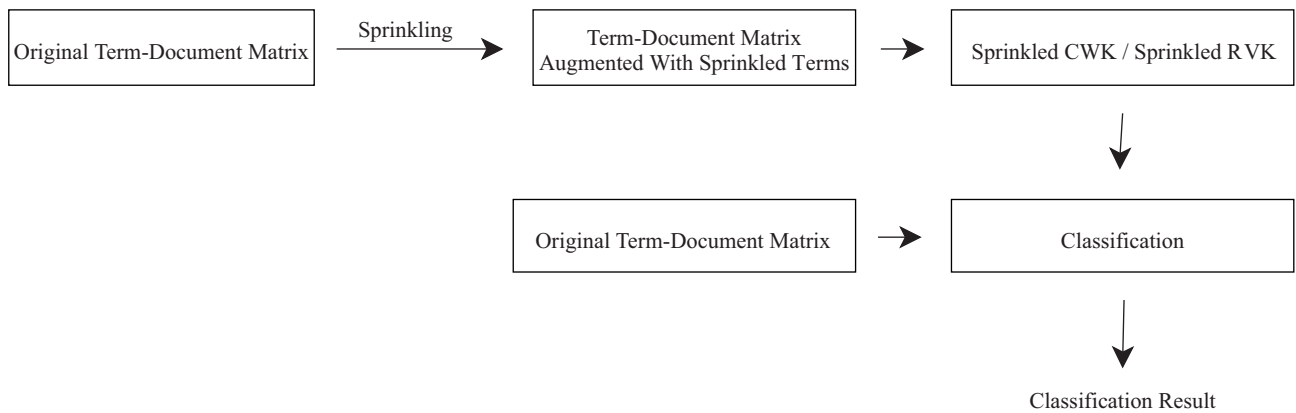


Figure 3. The architecture of sprinkled CWK and sprinkled RVK.

calculated from different formula the values are in different boundaries, respectively. Therefore, we normalize these two matrices separately using Eqs. (8) and (9):

$$\forall i, j \in \{1 \dots r\}, N_{gram_matrix_{CWK}}(d_i, d_j) = \frac{gram_matrix_{CWK}(d_i, d_j)}{max(gram_matrix_{CWK})}, \tag{8}$$

$$\forall i, j \in \{1 \dots r\}, N_{gram_matrix_{RVK}}(d_i, d_j) = \frac{gram_matrix_{RVK}(d_i, d_j)}{max(gram_matrix_{RVK})}, \tag{9}$$

where r is the number of documents in our corpus, $gram_matrix_{CWK}(d_i, d_j)$ is the gram (kernel) matrix generated from kernel function of CWK, $N_{gram_matrix_{CWK}}(d_i, d_j)$ is the normalized form of $gram_matrix_{CWK}(d_i, d_j)$, $gram_matrix_{RVK}(d_i, d_j)$ is the gram (kernel) matrix generated from kernel function of RVK, $N_{gram_matrix_{RVK}}(d_i, d_j)$ is the normalized form of $gram_matrix_{RVK}(d_i, d_j)$; respectively. After this normalization we combine these two gram matrices with a parameter λ , where λ is a positive real number ($0 < \lambda < 1$). In order to optimize λ , the following values are taken into consideration: $\{0.1, 0.25, 0.5, 0.75, 0.8, 0.9\}$. Combination of $N_{gram_matrix_{RVK}}$ and $N_{gram_matrix_{CWK}}$ matrices with parameter λ is given in Eq. (10):

$$Sim(d_i, d_j) = \lambda \times N_{gram_matrix_{CWK}} + (1 - \lambda) \times N_{gram_matrix_{RVK}} \tag{10}$$

$Sim(d_i, d_j)$ is the final similarity matrix between documents. $Sim(d_i, d_j)$ matrix is used as the kernel matrix in our composite model.

4.4. WSD using ensemble techniques

A majority-voting scheme is implemented in the classification phase to determine the output of the ensemble algorithm, which combines decisions of CWK, RVK, and their sprinkled versions.

5. Experiments and results

5.1. Datasets

In order to evaluate our algorithms we select the datasets, namely, hard and serve from SensEval. These datasets are commonly used datasets in the studies of WSD. Each occurrence of the words serve and hard was manually tagged with a WordNet sense. The number of different senses for serve and hard are four and three, respectively.

Serve data: There are 4378 instances with the verb serve in the serve dataset [49]. The verb sense has four different senses gathered from WordNet. These senses and their frequency distribution are shown in Table 3. This dataset has been extracted from the 1987 to 1989 Wall Street Journal corpus and the American Printing House for the Blind corpus.

Hard data: The hard dataset [49] contains 4333 instances taken from the San Jose Mercury News corpus. The adjective hard has three different senses gathered from WordNet. These senses and their frequency distribution are shown in Table 4. The distribution of instances is skewed in the "not easy (difficult)" sense; and the distribution of instances is equal in other two senses.

5.2. Experimental setup

In order to observe the behavior of our semantic kernels under different training set size conditions, the following percentage values for training set size is used: 5%, 10%, and 30%. Remaining documents are used for testing.

Table 3. Senses and frequencies in the serve dataset.

No	Sense	Frequency
1	Supply with food	1814
2	Hold an office	1272
3	<i>Function as something</i>	853
4	Provide a service	439

Table 4. Senses and frequencies in the hard dataset.

No	Sense	Frequency
1	not easy (difficult)	3455
2	not soft (metaphoric)	502
3	not soft (physical)	376

After running algorithms on 10 random splits for each of the training set ratios, we report average of these 10 results as in Tables 5–7. The main evaluation metric in our experiments is F-score, which is a popular choice of evaluation on datasets with skewed class distribution. All the proposed algorithms are implemented with Python⁵ using Python Data Analysis Library (pandas v0.22.0) and Machine Learning Libraries (scikit learn 0.19.1 and sklearn.ensemble). The experimental results are shown in Tables 5–7.

5.3. Evaluation results and discussions

Table 7 represents the F-score results of linear, CWK, sprinkled CWK, RVK, sprinkled RVK, composite, and ensemble kernels. According to Table 7, both CWK and sprinkled CWK have superior F-scores to baseline (linear kernel) at all training levels on the serve dataset.

Table 5. F-score on the hard dataset and the serve dataset.

Dataset	Train size(%)	Linear kernel	CWK	Sprinkled CWK	RVK	Sprinkled RVK	Composite kernel	Ensemble Kernel
Hard	5	43.00	45.77	45.75	48.40	44.81	53.06	45.10
	10	49.54	49.79	49.27	53.72	51.67	56.30	50.41
	30	61.68	58.44	59.99	62.16	59.28	64.50	56.60
Serve	5	50.65	59.89	59.12	55.04	55.02	59.64	59.31
	10	60.82	65.10	64.85	63.96	63.98	64.50	64.16
	30	72.89	74.71	73.40	74.28	74.25	74.58	74.75

⁵<https://www.python.org/downloads/>

For instance for the serve dataset; at training level 5%, F-score of CWK is 59.89% while F-score of linear kernel is 50.65%. The performance difference between RVK and linear kernel at training level 5% is 4.39%. For Hard dataset; the performance difference between RVK and Linear Kernel at training level 5% is 5.4%, which has a great importance since it is problematic to get labeled data in real-world cases.

Moreover, an ensemble algorithm which combines decisions of CWK, RVK, and their sprinkled versions yields better F-scores compared to individual CWK and RVK on the serve dataset at training level 30% as reported in Table 7. Furthermore, ensemble-based semantic kernel seems to be better than linear kernel at all training levels on the serve dataset.

Hard dataset behaves differently compared to the serve dataset under the methodology used. Composite kernels have better F-scores than all the other methods at all training levels. CWK and sprinkled CWK have close scores to the linear kernel. For instance at training level 10%, F-scores of linear kernel, CWK, RVK, and composite kernels are 49.54, 49.79, 53.72, and 56.30; respectively. According to the results, both CWK and sprinkled CWK produce somehow better classification accuracies compared to baseline at training level 5%.

Moreover, sprinkled CWK seems to generate higher classification accuracy than CWK at training level 30%. This can be explained by the reason that sprinkled features show their effect much more when there are more labeled data.

Table 6. Recall results on the hard dataset and the serve dataset.

Dataset	Train size(%)	Linear kernel	CWK	Sprinkled CWK	RVK	Sprinkled RVK	Composite kernel	Ensemble kernel
Hard	5	41.50	44.05	44.04	52.68	54.34	51.63	43.20
	10	46.47	45.67	45.66	55.87	54.45	57.11	43.94
	30	56.46	52.43	53.05	57.82	53.84	62.17	50.44
Serve	5	50.13	59.39	59.30	59.85	59.82	55.64	48.62
	10	58.62	67.35	64.16	66.22	66.23	59.66	53.22
	30	70.67	72.59	72.40	74.00	73.96	67.41	72.28

To completely evaluate the efficiency of our developed classification models, we also provide both precision and recall values in Tables 6 and 7, respectively. Table 6 shows the recall results of linear kernel, CWK, sprinkled CWK, RVK, sprinkled RVK, composite, and ensemble kernels. According to Table 6, for instance at training split 5% recall values of CWK, sprinkled CWK, RVK, sprinkled RVK, composite, and ensemble kernel are 44.05%, 44.04%, 52.68%, 54.34%, 51.63%, 43.20% while linear kernel is 41.50% on the hard dataset, respectively.

Furthermore, the precision values of linear kernel, CWK, sprinkled CWK, RVK, sprinkled RVK, composite, and ensemble kernels are reported in Table 7. According to Table 7, the greatest precision value at training split 30% on the hard dataset is achieved with ensemble kernel: it reaches 80.86% precision score while linear kernel reaches 78.55% precision score at the same training split. The same trend can be noticed on the serve dataset at training split 30% : Ensemble kernel reaches 78.14% precision score while linear kernel reaches 77.42% precision score at the same training split.

6. Conclusions and future directions

The experimental results show that our WSD algorithms can be configured to have high F-score compared to linear kernel. One of the conclusions driven from this study is that class-based semantic values in a kernel setting improves the classification accuracy in WSD. Secondly, sprinkled features seems to help in classification

Table 7. Precision results on the hard dataset and the serve dataset.

Dataset	Train size(%)	Linear kernel	CWK	Sprinkled CWK	RVK	Sprinkled RVK	Composite kernel	Ensemble kernel
Hard	5	79.97	69.92	69.90	56.25	50.25	59.94	69.59
	10	79.37	72.87	72.86	57.93	59.92	60.05	74.63
	30	78.55	77.68	76.58	72.18	75.69	69.95	80.86
Serve	5	70.74	69.17	64.17	62.87	62.82	60.41	59.14
	10	73.52	74.66	74.15	68.68	68.70	65.54	65.90
	30	77.42	76.91	76.05	76.52	76.58	70.86	78.14

especially where there is much labeled data. Furthermore, we observe that composition of CWK and RVK gives higher classification accuracy than individual CWK and RVK on the hard dataset. Moreover, we also observe that an ensemble algorithm which combines decisions of CWK, RVK, and their sprinkled versions yields better F-scores compared to individual CWK and RVK on the serve dataset at training level 30%. As a future work, we aim to perform more experiments on other larger-scale WSD corpora. We will also analyze how the suggested sprinkled methodologies improve classification accuracy in WSD in some test cases. An additional item on our agenda is to expand our approaches by adding further semantic information. Moreover, the promising results of semantic kernels indicate that using such kernels with other known kernelized algorithms such as kernel PCA for WSD or kernelized Gaussian mixture models might give us improved results when dealing with WSD problems.

Acknowledgments

The authors would like to thank Bektaş İLDEM and Ece ÖZEN for their help in writing this paper. This work is supported in part by The Scientific and Technological Research Council of Turkey (TÜBİTAK) grant number 116E047 and grant number 118E315. Some parts of this work is done during the postdoctoral studies of the third author at Marmara University as a part of TÜBİTAK project grant number 116E047. Points of view in this document are those of the authors and do not necessarily represent the official position or policies of TÜBİTAK.

References

- [1] Joshi M. Kernel methods for word sense disambiguation and abbreviation expansion in the medical domain. MSc, University of Minnesota, MN, USA, 2006.
- [2] Shawe-Taylor J, Nello C. Kernel Methods for Pattern Analysis. London, UK: Cambridge University Press, 2004.
- [3] Giuliano C, Gliozzo A, Strapparava C. Kernel methods for minimally supervised WSD. Computational Linguistics 2009; 35 (4): 513-528. doi: 10.1162/coli.2009.35.4.35407
- [4] Magnini B, Strapparava C, Pezzulo G, Gliozzo A. The role of domain information in word sense disambiguation. Natural Language Engineering 2002; 8 (4): 359-373. doi: 10.1017/S1351324902003029
- [5] Cancedda N, Gaussier E, Goutte C, Renders J-M. Word-sequences kernels. Journal of Machine Learning Research 2003; 3 (2): 1059-1082. doi: 10.14257/jmlrs.2003.10.1.08
- [6] Bloehdorn S, Hotho A. Boosting for text classification with semantic features. In: ACM 2004 International Conference on Knowledge Discovery and Data Mining; WA, USA; 2004. pp. 70-87.

- [7] Scott S, Matwin S. Feature engineering for text classification. In: Morgan Kaufmann 1999 International Conference on Machine Learning; Bled, Slovenia; 1999. pp. 379-388.
- [8] Agirre E, Rigau G. A proposal for word sense disambiguation using conceptual distance. In: John Benjamins Bv 1995 Recent Advances in NLP; Tzigrav Chark, Bulgaria; 1995. pp. 258-264.
- [9] Banerjee S, Pedersen T. Extended gloss overlaps as a measure of semantic relatedness. In: Morgan Kaufmann 2003 International Joint Conference on Artificial Intelligence; Acapulco, Mexico; 2003. pp. 805-810.
- [10] Moschitti A. A study on convolution kernels for shallow statistic parsing. In: ACL 2004 Annual Meeting of the Association for Computing Linguist; Barcelona, Spain; 2004. pp. 335-342.
- [11] Zhao S, Grishman R. Extracting relations with integrated information using kernel methods. In: ACL 2005 Annual Meeting of the Association for Computing Linguist; Ann Arbor, MI, USA; 2005. pp. 419-426.
- [12] Cristianini N, Shawe-Taylor J, Lodhi H. Latent semantic kernels. *Journal of Intelligent Information Systems* 2004; 18 (2-3): 127-152. doi: 10.1023/A:1013625426931
- [13] Miller G, Beckwith R, Fellbaum C, Gross D, Miller K. *Five Papers of WordNet*. Princeton, NY, USA: Cognitive Science Laboratory Press, 1993.
- [14] Altinel B, Diri B, Ganiz MC. A novel semantic smoothing kernel for text classification with class-based weighting. *Knowledge-Based Systems* 2015; 89 (11): 265-277. doi: 10.1016/j.knosys.2015.07.008
- [15] Altinel B, Ganiz MC, Diri B. Instance labeling in semi-supervised learning with meaning values of words. *Engineering Applications of Artificial Intelligence* 2017; 62 (6): 152-163. doi: 10.1016/j.engappai.2017.04.003
- [16] Biricik G, Diri B, Sönmez AC. Abstract feature extraction for text classification. *Turkish Journal of Electrical Engineering & Computer Sciences* 2012; 20 (1): 1137-1159. doi:10.3906/elk-1102-1015
- [17] Lan M, Tan CL, Su J, Lu Y. Supervised and traditional term weighting methods for automatic text categorization. *Transactions on Pattern Analysis and Machine Intelligence* 2008; 31 (4): 721-735. doi: 10.1109/TPAMI.2008.110
- [18] Bloehdorn S, Moschitti A. Combined syntactic and semantic kernels for text classification. In: Springer 2007 Advances in Information Retrieval Conference; Rome, Italy; 2007. pp. 307-318.
- [19] Bloehdorn S, Basili R, Cammisa M, Moschitti A. Semantic kernels for text classification based on topological measures of feature similarity. In: IEEE 2006 International Conference on Data Mining; Hong Kong, China; 2006. pp. 808-812.
- [20] Nasir JA, Karim A, Tsatsaronis G, Varlamis I. A knowledge-based semantic kernel for text classification. In: Springer 2011 International Conference on String Processing and Information Retrieval; Pisa, Italy; 2011. pp. 261-266.
- [21] Nasir JA, Varlamis I, Karim A, Tsatsaronis G. Semantic smoothing for text clustering. *Knowledge-Based Systems* 2013; 54 (12): 216-229. doi: 10.1016/j.knosys.2013.09.012
- [22] Siolas G, d'Alché-Buc F. Support vector machines based on a semantic kernel for text categorization. In: IEEE 2000 International Joint Conference on Neural Networks; Como, Italy; 2000. pp. 205-209.
- [23] Wang P, Domeniconi C. Building semantic kernels for text classification using wikipedia. In: ACM 2008 International Conference on Knowledge Discovery and Data Mining; Las Vegas, NV, USA; 2008. pp. 713-721.
- [24] Altinel B, Ganiz MC, Diri B. A novel higher-order semantic kernel for text classification. In: IEEE 2013 International Conference on Electronics, Computer and Computation; Ankara, Turkey; 2013. pp. 216-219.
- [25] Altinel B, Ganiz MC, Diri B. A semantic kernel for text classification based on iterative higher order relations between words and documents. In: Springer 2014 International Conference on Artificial Intelligence and Soft Computing; Zakopane, Poland; 2014. pp. 505-517.
- [26] Altinel B, Ganiz MC, Diri B. A simple semantic kernel approach for SVM using higher-order paths. In: IEEE 2014 International Symposium on Innovations in Intelligent Systems and Applications; Alberobello, Italy; 2014. pp. 431-435.

- [27] Wang T, Rao J, Hu Q. Supervised word sense disambiguation using semantic diffusion kernel. *Engineering Applications of Artificial Intelligence* 2014; 27 (1): 167-174. doi: 10.1016/j.engappai.2013.08.007
- [28] Wang T, Li W, Liu F, Hua J. Sprinkled semantic diffusion kernel for word sense disambiguation. *Engineering Applications of Artificial Intelligence* 2017; 64 (9): 43-51. doi: 10.1016/j.engappai.2017.05.010
- [29] Chakraborti S, Lothian R, Wiratunga N, Watt S. Sprinkling: supervised latent semantic indexing. In: Springer 2006 European Conference on IR Research; London, UK; 2006. pp. 510-514.
- [30] Valentini G, Masulli F. Ensembles of learning machines. In: Springer 2002 Italian Workshop on Neural Nets; Vietri sul Mare, Italy; 2002. pp. 3-20.
- [31] Freund Y, Schapire RE. Experiments with a new boosting algorithm. In: Morgan Kaufmann 1996 International Conference on Machine Learning; Bari, Italy; 1996. pp. 148-156.
- [32] Bauer E, Kohavi R. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning* 1999; 36 (1-2): 105-139. doi: 10.1023/A:1007515423169
- [33] Dietterich TG. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning* 2000; 40 (2): 139-157. doi: 10.1023/A:1007607513941
- [34] Kittler J, Hatef M, Duin RPW, Matas J. On combining classifiers. *Transactions on Pattern Analysis and Machine Intelligence* 1998; 20 (3): 226-239. doi: 10.1109/34.667881
- [35] Allwein EL, Schapire RE, Singer Y. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research* 2000; 1 (12): 113-141. doi: 10.1162/15324430152733133
- [36] Freund Y, Schapire RE. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 1997; 55 (9): 119-139. doi: 10.1006/jcss.1997.1504
- [37] Polikar R. Ensemble based systems in decision-making. *Circuits and Systems Magazine* 2006; 6 (3): 21-45. doi: 10.1109/MCAS.2006.1688199
- [38] Kimura F, Shridhar M. Handwritten numerical recognition based on multiple algorithms. *Pattern Recognition* 1991; 24 (10): 969-983. doi: 10.1016/0031-3203(91)90094-L
- [39] Perrone MP, Cooper LN. When networks disagree: ensemble methods for hybrid neural networks. In: Mammone R (editor). *Artificial Neural Networks for Speech and Vision*. NY, USA: Chapman & Hall Press, 1993, pp. 126-142
- [40] Suen CY, Lam L. Multiple classifier combination methodologies for different output levels. In: Springer 2000 Multiple Classifier Systems; Cagliari, Italy; 2000. pp. 52-66.
- [41] Cho SB, Kim JH. Combining multiple neural networks by fuzzy integral for robust classification. *Transactions on Systems, Man, and Cybernetics* 1995; 25 (2): 380-384. doi: 10.1109/21.364825
- [42] Breiman L. Bagging predictors. *Machine Learning* 1996; 24 (2): 123-140. doi: 10.1007/BF00058655
- [43] Magnini B, Cavaglia G. Integrating subject field codes into WordNet. In: IEEE 2000 International Conference on Language Resources and Evaluation; Athens, Greece; 2000. pp. 1413-1418.
- [44] Gliozzo A, Giuliano C, Strapparava C. Domain kernels for word sense disambiguation. In: ACL 2005 Annual Meeting of the Association for Comput. Linguist; Ann Arbor, MN, USA; 2005. pp. 403-410.
- [45] Bruce R, Wiebe J. A new approach to word sense disambiguation. In: ACL 1994 ARPA Workshop on Human Language Technology; Plainsboro, NJ, USA; 1994. pp. 244-249
- [46] Yarowsky D. One sense per collocation. In: ACL 1993 Workshop on Human Language Technology; NJ, USA; 1993: pp. 266-271.
- [47] Mavroudis D, Tsatsaronis G, Vazirgiannis M, Theobald M, Weikum G. Word Sense Disambiguation for Exploiting Hierarchical Thesauri in Text Classification. In: Jorge AM, Torgo L, Brazdil P, Camacho R, Gama J, editors. Berlin, Germany: Lecture Notes in Computer Science book series press, 1995, pp. 181-192.
- [48] Alpaydm E. *Introduction to Machine Learning*. Cambridge, MA, USA: MIT press, 2004.
- [49] Leacock C, Miller GA, Chodorow M. Using corpus statistics and WordNet relations for sense identification. *Computational Linguistics* 1998; 24 (1): 147-165. doi: 10.3115/1075812.1075867