REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

CORPUS-BASED SEMANTIC KERNELS FOR SUPERVISED AND SEMI-SUPERVISED TEXT CLASSIFICATION

BERNA ALTINEL

PhD. THESIS
DEPARTMENT OF COMPUTER ENGINEERING
PROGRAM OF COMPUTER ENGINEERING

ADVISER
ASSOC. PROF. DR. BANU DİRİ

CO-ADVISER
ASSIST. PROF. DR. MURAT CAN GANİZ

İSTANBUL, 2015

REPUBLIC OF TURKEY

YILDIZ TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

CORPUS-BASED SEMANTIC KERNELS FOR SUPERVISED AND SEMI-SUPERVISED TEXT CLASSIFICATION

A thesis submitted byBerna ALTINELin partial fulfillment of the requirements for the degree of **DOCTOR OF PHILOSOPHY**is approved by the committee on 30.12.2015 in Department of Computer Engineering, Computer Engineering Program.

Thesis Adviser

Assoc. Prof. Dr. Banu DİRİ
Yıldız Technical University

Co- Adviser

Assist. Prof. Dr. Murat Can GANİZ
Marmara University

Approved By the Examining Committee

Assoc. Prof. Dr. Banu DİRİ
Yıldız Technical University______________________

Assist. Prof. Dr. Murat Can GANİZ
Marmara University ______________________

Assist. Prof. Dr. M. Fatih AMASYALI, Member
Yıldız Technical University ______________________

Assist. Prof. Dr. Zeynep ORHAN, Member
Fatih University ______________________

Prof. Dr. Zehra ÇATALTEPE, Member
İstanbul Technical University ______________________

Assoc. Prof. Dr. Elif KARSLIGİL, Member
Yıldız Technical University ______________________

Assist. Prof. Dr. Arzucan ÖZGÜR, Member
Boğaziçi University ______________________

This study was supported by the Scientific and Technological Research Council of Turkey (TUBITAK) Grant No: 111E239.

ACKNOWLEDGEMENTS

I am deeply thankful to my co-adviser,Assist. Prof. Dr. Murat Can Ganiz, forhis valuable support and mentorship during my thesis journey. With the help of his knowledge and guidance, I was able to look at my research from different viewpoints. It would have been very difficult to complete this study without hissupport. I am very grateful him for the big effort in teaching and directing me in this research. I really owe him a lot for the enormous amount of time spent with me discussing different problems ranging from theoretical issues down to technical details. I also would like to thank to my adviser, Assoc. Prof. Dr. Banu Diri, for her support and mentorship during this study. She did not only help me during my research but encouraged me to make it better. She also gives me precious advice as well as the moral to overcome difficulties.

I would like to thank my committee members; Assist. Prof. Dr. M.Fatih Amasyalı and Assist. Prof. Dr. Zeynep Orhan, for their support and contribution to my thesis. They were always available to answer my questions and encourage me during this tough work.

I also would like to thank to my previous advisers Prof. Dr. Zehra Çataltepe who is the adviser of my MSc thesis and Assoc. Prof. Dr. Ender Özcan who is the adviser of my BSc thesis. I am very grateful to them for directing and motivating me for the next step in academic life and always support me even after my graduation.

I also would like to Prof. Dr. A. Coşkun Sönmez whowas one of the most wonderful and helpful professors that I have ever seen. He was always positive and used to be at student side and support his students. Even after his death, I am sure that our love for him will not be less; because love and respect are actually immortal in life.

I thank very much all of my colleagues at Marmara University for their patient understating of my overwhelming schedule.

I would like to thank all of my dearest friends who always support me and keep joy in my life for the last fifteen years.

I also owe Ali a lot;because hehas been much safer, better and meaningful side of my life for about eight years.

Finally, I would like to thank my family. All members of my family; my father Nurettin, my mother Metine and my brother Baran are separately very important for me. I am grateful them for the big support in my life. It would have been very hard to complete this and my all previous studies without their unconditional and endless help and support. They are always nearby me. I really need to thank to them for their support for encouraging me and giving their full support with my every decision. I am very lucky to have such a wonderful family.

This thesis is dedicated to my family, who has taken care of me without expecting any return.

December, 2015

BernaALTINEL

TABLE OF CONTENTS

LIST OF SYMBOLS

a	Number of documents in the positive category which containterm w
b	Bias
B	Length of a section (paragraph, class, etc.) in words
c	Number of documents in the negative category which contain term w
C	Regularization parameter (misclassification cost parameter)
$\lvert C \rvert$	Number of classes
c_{fw}	Number of classes which contains term w
C_m	Counts m-tuple of the elements of *Sw* appears in the same document
$\lvert di \rvert$	The length of the document
d_p	Term vector of document *p* which shows terms with their frequency
d_{fw}	Number of documents which contains term w
D	Term by document matrix of the corpus
D^T	Transpose of *D* matrix
$E(x)$	Expectation value
F	First-order path matrix
FN	Normalized first-order paths matrix
G	Generator that displays the initial semantic similarities between words
$G_{p,q}$	Gram matrix shows the kernel value between documents *dp* and *dq*
h_i	Initial learner
L_o	Original labeled instances
L_p	Previously unlabeled instances with their current predicted labels
L	Total of $L_o$and L_p
$L_{labeled}$	Labeled training examples
L_d	Length of a document
$M_{labeled}$	Matrix shows the meaningfulness of the terms in the labeled set for each class
m_{ij}	Occurrence frequency of the j^{th}word in the ith document;
m_i	Row vector representing the document *i*
m_j	Column vector corresponding to word *j*
N	Total number of documents in the corpus
N_w	Total number of documents contain term *w*
N	Total number of documents in the training set
NR	Row normalization matrices
NC	Column normalization matrices
P	Probability

p_{td}	Equals the number of times that *t* occurs in *d* divided by the total number of times that *t* occurs
S	Second-order path matrix
$S_{i,j}$	Semantic smoothing matrix shows the relatedness between words *i* and *j*
SN	Normalized second-order paths matrix
SR	Row (document) similarity matrix
SC	Column (word) similarity matrix
S_w	The set of all words in *N* documents
t_i	Term
tf_w	Term frequency of word *w*
$tf_{c\ w,c}$	Total term frequency of word *w* in the documents of class *c*
$tf_{\ w,d}$	Total term frequency of word *w* in the document *d*
U	Unlabeled examples
W	Weight Matrix for each word *w* and class *k*
w	Weight vector
$\phi(d_1)$	Transformation of document
α_i	Langlier's multiplier
α	Significance level in student's t-test
$\bar{\phi}$	Mapping from an input space into a feature space.
ξ	Vector of slack variables
λ	Decay factor

LIST OF ABBREVIATIONS

1A1	One-Against-One
1AA	One-Against-All
1D	One Dimensional
BOW	Bag-of-Words
CBT	Corpus-Based Thesaurus
CMK	Class Meaning Kernel
CWK	Class Weighting Kernel
EM	Expectation–maximization
HONB	Higher-Order Naive Bayes
HOSK	Higher-Order Semantic Kernel
HOTK	Higher-Order Term Kernel
IG	Information Gain
IHOSK	Iterative Higher-Order Semantic Kernel
ILBOM	Instance Labeling Based on Meaning
IPL	Inverted Path Length
IR	Information Retrieval
k-NN	K-Nearest Neighbor
LSA	Latent Semantics Analysis
LSO	Lowest Super Ordinate
NB	Naive Bayes
NFA	Number of False Alarms
RBF	Radial Basis Function
QP	Quadratic Programming
SMO	Sequential Minimal Optimization
SO-CMK	Second-Order Class Meaning Kernel
SO-CWK	Second-Order Class Weighting Kernel
SSL	Semi-supervised Learning
SSTK	Semantic Syntactic Tree Kernel
SVM	Support Vector Machine
TF	Term Frequency
TF-ICF	Term Frequency-Inverse Class Frequency
TF – IDF	Term Frequency Inverse Document Frequency
TF-RF	Term Frequency-Relative Frequency
TS	Training set percentages
TSVM	Transductive Support Vector Machine
UMK	Unsupervised Meaning Kernel

VS	Vector Space
VSM	Vector Space Model
WKNN	Weighted K-Nearest Neighbor Method
WSD	Word Sense Disambiguation
WWW	World Wide Web

LIST OF FIGURES

Page

LIST OF TABLES

Page

ABSTRACT

CORPUS-BASED SEMANTIC KERNELS FOR SUPERVISED AND SEMI-SUPERVISED TEXT CLASSIFICATION

Berna ALTINEL

Computer Engineering Department

Ph.D. Thesis

Adviser: Assoc. Prof. Dr. Banu DİRİ

Co-Adviser: Assist. Prof. Dr. Murat Can GANİZ

Text categorization plays a crucial role in both academic and commercial platforms due to the growing demand for automatic organization of documents. Kernel-based classification algorithms such as Support Vector Machines (SVM) have become highly popular in the task of text mining. This is mainly due to their relatively high classification accuracy on several application domains as well as their ability to handle high dimensional and sparse data which is the prohibitive characteristics of textual data representations. Recently, there is an increased interest in the exploitation of background knowledge such as ontologies and corpus-based statistical knowledge in text categorization. It has been shown that, by replacing the standard kernel functions such as linear kernel with customized kernel functions which take advantage of this background knowledge, it is possible to increase the performance of SVM in the text classification domain. Based on this, we developed a variety of semantic kernel methods in order to explore the capabilities of higher-order paths, class-based meaning values and class-based weighting of terms in both supervised learning and SSL setting for SVM.

We propose several corpus-based semantic kernels which implicitly extract and make use of semantic relations such as Higher-Order Semantic Kernel (HOSK), Iterative Higher-Order Semantic Kernel (IHOSK) and Higher-Order Term Kernel (HOTK) for SVM. HOSK makes use of higher-order co-occurrence paths of terms between documents. In HOSK, the simple dot-product between feature vectors of the documents consist of term frequencies yields a first-order document relation matrix (F). Second–order document matrix (S) is formed by multiplying F with itself. S is used as kernel matrix in HOSK's transformation from input space into feature space. The experimental results show that HOSKshows an improvement on accuracy over linear kernel.A more advanced model is IHOSK which uses higher-order paths between documents and terms together in an iterative form. The document similarity matrix is produced iteratively using SR (a similarity matrix between documents) and SC (a similarity matrix between terms). Experiment results show that the classification performance increases relative to the linear kernel. In our following study, we consider less complex higher-order kernel, HOTK that is based on higher-order paths between terms only. HOTK is much simpler than IHOSK and also requires less computational resources.

We also propose a novel approach for building a semantic kernel for SVM, which we name CMK. We applied CMK in a Semi-supervised Learning (SSL) setting with an addition of a new approach to initial labeling of unlabeled data, called ILBOM. The suggested approaches smooth the term weights of a document in BOW representation by class-based meaning values of terms. These approaches reduce the disadvantages of BOW by increasing the importance of class specific concepts which can be synonymous or closely related in a class. The meaning values of terms are calculated according to the Helmholtz principle from Gestalt theory in the context of classes. Our experimental results show that both CMK and ILBOM widely outperform the classification accuracy of the linear kernel.

Additionally we also propose another approach which is called Class Weighting Kernel (CWK). This approach is similar to CMK however it provides an improvement over CMK in terms of mainly the calculation time. This class-based weighting basically groups terms based on their importance for each class. Therefore it smooths the representation of documents which changes the orthogonality of the vector space model by introducing class-based dependencies between terms.

The main contribution of this dissertation is building novel semantic kernels those are applied to supervised and semi-supervised text classification.We show that kernels performing much better than standard kernels in terms of classification accuracy. The proposed approaches have independency of the outside semantic sources such as WordNet, so that they can be applied to any language domain. They also form a foundation that can easily be combined with other term-based semantic similarity methods such as unsupervised semantic similarity measures. To the best of our knowledge, higher-order paths and class-based values of terms are used in the transformation phase of SVM for the first time in the literature and give significant benefits on the semantic smoothing of terms in a kernel for text classification.

Key Words: Support Vector Machines, text classification, semantic smoothing kernel, supervisedkernel, higher-order co-occurrence, higher-order paths, Helmholtz principle, class-based term weighting.

YILDIZ TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

ÖZET

EĞİTİCİLİ VE YARI-EĞİTİCİLİ METİN SINIFLANDIRMASI İÇİN DERLEM TABANLI ANLAMBİLİMSEL ÇEKİRDEKLER

Berna ALTINEL

Bilgisayar Mühendisliği Bölümü

Doktora Tezi

Tez Danışmanı: Doç. Dr. Banu DİRİ

Tez Eş-Danışmanı : Yrd. Doç. Dr. Murat Can GANİZ

Metin sınıflandırma, belgelerin otomatik organizasyonu için artan talepten ötürü hem akademik hem de ticari platformlarda önemli bir rol oynamaktadır.Destek Vektör Makineleri (SVM) gibi çekirdek temellisınıflandırma algoritmaları metin madenciliği görevinde son derece popüler hale gelmişlerdir. Bu durum esas olarak SVM'in çeşitli uygulama alanları üzerindeki nispeten yüksek sınıflandırma doğruluğunun yanı sıra yüksek boyutlu ve seyrek veriyi işlemeyebilme yeteneklerindende kaynaklanmaktadır.Son zamanlarda, metin sınıflandırmasındaontolojiler ve derlem temelli istatistiki bilgi gibi arka plan bilgi birikiminden yararlanmaya yönelik artan bir ilgi söz konusudur.Doğrusal çekirdek gibi standart çekirdek fonksiyonları yerine bu arka plan bilgisinin avantajlarından faydalanan özelleştirilmiş çekirdek fonksiyonlarınıkullanarak SVM'in metin sınıflandırma alanındaki performansını arttırmanın mümkün olduğu gösterilmiştir.Buna dayanarak,SVM için eğiticili ve yarı-eğiticili anlambilimsel düzeltme çekirdeklerinde, daha yüksek mertebeden yolların, terimlerin sınıf temelli anlamsal değerlerinin ve sınıf temelli ağırlık değerlerinin yeteneklerini keşfetmek amacıyla çeşitli yöntemler geliştirilmiştir.

Bu çalışamda Yüksek MertebedenAnlambilimsel Çekirdek (HOSK), Özyineli Yüksek Mertebeden Anlambilimsel Çekirdek (IHOSK) ve Yüksek Dereceden Terim Çekirdeği (HOTK) gibi dolaylı anlambilimsel ilişkileri çıkartan ve kullanan derlemtemelli çeşitli anlambilimsel çekirdekler önerilmiştir.HOSKterimlerin belgeler arasındaki yüksek

mertebeden yolları kullanır. HOSK'ta belgelerin özellik vektörleri arasındaki basit iç çarpım sonucunda birinci dereceden bir matris (F) elde edilir.HOSK belgeler, bu özellik vektörleri arasında basit nokta ürünün birinci dereceden bir matris (F) elde edilir. İkinci dereceden eş-oluşum matrisi (S), F'nin kendisi ile çarpılması sonucu oluşturulur. S, HOSK'un giriş uzayından özellik uzayına dönüşümündeki çekirdek matrisi olarak kullanılmaktadır. Deneysel sonuçlar HOSK'un doğrusal çekirdek üzerindedoğruluk açısından bir iyileştirme sağladığını göstermektedir. HOSK'un daha gelişmiş bir modeli debelgeler ve terimler arasındaki yüksek dereceli yolları yinelemeli bir şekilde kullanan IHOSK'tur.Belgeler ve terimler arasındakianlambilimsel ilişki;belgeler arasındaki benzerlik matrisini terimler arasındaki benzerlik matrisini kullanarak ve terimler arasındaki benzerlik matrisini de belgeler arasındaki benzerlik matrisini kullanarak hesaplayan ve χ-Sim olarak adlandırılan özyineli bir tekniktenuyarlanmıştır. Belge benzerlik matrisi, SR (belgeler arası benzerlik matrisi) ve SC (terimler arası benzerlik matrisi) kullanılarak özyineli bir şekilde üretilir. Deney sonuçları sınıflandırma performansının doğrusal çekirdeğe kıyasla daha da arttığını göstermektedir. Bir sonraki çalışmamızda, daha az karmaşıklıkta yüksek-mertebeli çekirdekler düşünülmüştür; HOTK sadece terimler arasındaki yüksek-mertebeli yollara bağlıdır.HOTK'deki anlambilimsel çekirdek dönüşümü sadece eğitim kümesindeki terimler arası yüksek-mertebeli eş-oluşumlar kullanılarak yapılır. HOTK, IHOSK'dan daha basittir ve aynı zamanda daha az hesaplama kaynakları gerektirir.

Bu çalışmada, SVM için anlambilimsel çekirdek inşa eden CMK olarak adlandırılanyeni bir yaklaşım önerilmektedir. CMK'yı başlangıçtaki etiketsiz veriyi etiketlendiren yeni bir yöntem eklentisi ile yarı-eğiticili öğrenmeye uyguladık ve bunu ILBOM olarak adlandırdık. Önerilen yaklaşımlar bir belge içindeki BOW ile temsil edilen terimlerin ağırlıklarını,terimlerin sınıf temelli anlamsal değerlerini kullanarak düzeltmektedir. Bu da sınıflar üzerinde ayırt ediciliği olmayan genel amaçlı kullanılan terimlerin önemini azaltırken,önemli ya da başka bir deyişle anlamlı terimlerin önemini artırmaktadır. Bu yaklaşımlar, eşanlamlı terimler ya da sınıfla yakından ilgili terimler gibi sınıfa özgü kavramların önemini arttırarak BOW'un dezavantajlarını azaltmaktadır. Terimlerin sınıflar bağlamındaki anlamsal değerleri Gestalt teoriden Helmholtz esasına göre hesaplanmaktadır. Deneysel sonuçlarımız CMK ve ILBOM'un doğrusal çekirdekten daha üstün bir sınıflandırma keskinliğisağladığını göstermektedir.

Ayrıca Sınıf Ağırlıklı Çekirdek (CWK) olarak adlandırılan başka bir yaklaşım da bu çalışmada önerilmiştir. Bu yöntem CMK'ya benzemektedir ancak; CWK özellikle hesaplama zamanı konusunda bir gelişme sağlamaktadır. Temelde bu sınıf temelli ağırlıklandırma her sınıf için terimleri önemlilik durumlarına göregruplandırır. Bu nedenle bu sınıf temelli ağırlıklandırmabelgelerin gösterimini düzeltir ki, bu da terimler arasına sınıf temelli bağımlılıklar getirerek vektör uzayı modelinin dikliğini değiştirir. Sonuç olarak, istisnai durumlarda, hiç ortak terim içermedikleri halde eğer belirli bir sınıf için benzer şekilde ağırlıklandırılmış iki belge benzer görülebilir.

Bu tezin temel katkısı standart çekirdeklerden çok daha iyi sınıflandırma doğruluğu sergileyebilen çözümler geliştirilmesi olarak düşünülebilir. Önerilen yaklaşımların ikinci katkısıbu modellerin WordNet gibi dış anlambilimsel kaynaklardan bağımsız olmaları ve bu sebepten ötürü herhangi bir dile uygulanabilir olmalarıdır. Bizim yöntemlerimizin diğer bir katkısı da eğiticisiz anlambilimsel benzerlik ölçümleri gibi diğer terim temelli anlambilimsel benzerlik yöntemleri ile kolayca kombine edilebilir bir yapıya temel oluşturmalarıdır. Yöntemlerimizin özellikle sınıf bazlı yöntemlerimizi başka bir avantajı da, bunların yürütüm süresi ile ilgilidir. Bizim bilgimize göre, yüksek dereceli yollar ve terimlerin

sınıf temelli değerleri SVM'in dönüşüm aşamasında ilk kez kullanılmaktadırve metin sınıflandırma için bir çekirdekte terimlerin anlambilimsel olarakdüzeltilmesi üzerine önemli bir bakış açısı kazandırabilir.

Anahtar Kelimeler: Destek Vektör Makineleri, metin sınıflandırma, anlambilimsel düzeltmeli çekirdek, eğiticili çekirdek, yüksek-mertebeli eş-oluşum, yüksek-mertebeli yollar, Helmholtz presibi, sınıf-temelli terim ağırlıklandırma.

CHAPTER 1

INTRODUCTION

1.1 Literature Review

In recent years, with the ever accumulating online information on the Internet and social media, text categorization has become one of the key techniques for organizing and handling textual data. Automatically processing these huge amounts of textual data is an essential problem. Text classification can be defined as the utilization of a supervised learning methodology to assign predefined class labels to documents using a model learned from labels of the documents in the training set. An important requirement of efficient and accurate text classification systems is to organize documents into pre-determined categories that contain similar documents. In order to achieve this goal, several classification algorithms depend on similarity or distance measures that compare pairs of text documents similarity. It is also known that vector space representation of textual documents yields high dimensionality and related to this; sparsity. This is especially a problem when there are a large number of category labels but limited amount of training data. It is thus crucial that a good text classification algorithm should scale well with the increasing number of features and classes. Most importantly, words in textual data carry semantic information, i.e., the sense carried by the terms of the documents. Consequently, an ideal text classification algorithm should be able to make use of this semantic information.

Bag-of-words (BOW) feature representation is well accepted as the fundamental approach in the domain of text classification. In BOW approach documents are characterized by the frequencies of individual words or terms that occur in the document and each term represents a dimension in a vector space independent of other terms in the same document[1]. It basically focuses on the frequency of words. The BOW approachover simplifies the representation of terms in documents by ignoring the several different syntactic or semantic relations between terms in natural language, e.g. it treats polysemous words (words with multiple meanings) as a single entity. For instance the term “bank” may have different meanings like financial institution or a river side based on the context it appears. Additionally, the BOW feature representation maps synonymous words into different components [1]. In principle, as Steinbach et al. [3] investigate, each class of documents has two kinds of vocabulary: one is “core” vocabulary which are intimately associated to the topic of that class, the other type is “general” vocabulary (e.g. stop words) those may have similar distributions on different classes. Therefore, two different documents from different classes may share many general words and will have high similarity based on their BOW representations.

In order to address these problems several methods have been proposed which use a measure of relatedness between term on Word Sense Disambiguation (WSD), Text Classification and Information Retrieval (IR) domains. Semantic relatedness computations fundamentally can be categorized into three such as knowledge-based systems, statistical approaches and hybrid methods which combine both ontology-based and statistical information[4]. Knowledge-based systems use a thesaurus or ontology to enhance the representation of terms by taking advantage of semantic relatedness among terms, for examples see[5], [6], [7], [8],[9], [10],[11] and [2]. For instance in [5]and [11] the distance between words in WordNet [12] is used to capture semantic similarity between English words. The study in [5] uses super-concept declaration with different distance measures between words from WordNet such as Inverted Path Length (IPL), Wu-Palmer Measure, Resnik Measure and Lin Measure. A recent study of this kind can be found in [13], which uses HowNet as a Chinese semantic knowledge-base. The second type of semantic relatedness computations between terms are corpus-based systems in which some statistical analysis based on the relations of terms in the set of training documents is performed in order to reveal latent similarities between them [44]. One of the famous corpus-based systems is Latent Semantics Analysis (LSA) [45] that partially solves the synonymy problem. Finally, approaches of the last category are called hybrid since they combine the information acquired both from the ontology and the statistical analysis of the corpus [9] and[92]. There is a recent survey in [44] about these studies.

1.2 Aims of the Dissertation

The aim of this work is to develop semantic kernels which make use of implicit semantic relations to improve the accuracy of kernel based classifier such as SVM. The proposed approaches explore the capabilities of higher-order paths and class-based meaning and class-based term weights in both supervised learning and SSL settings for SVM. The suggested approaches smooth the terms of a document in BOW representation (document vector represented by term frequencies) by higher-order paths between both documents and terms, class-based meaning values of terms and class-based term weights. Our target is to capture latent semantic information between the terms and between the documents. It is important to note that SVM with linear kernel is one the state of the art algorithms for text classification [19], [20]. This traditional kernel can be considered as a first-order method since its context is a single document and it model just the first-order co-occurrences of the terms. However, higher-order kernels make use of the higher-order paths that include several different terms and documents in the context of the whole dataset. Furthermore, class-based kernels increase the importance of significant or in other words meaningful terms for each class while reducing the importance of general terms which are not useful for discriminating the classes. Theseproposed approaches reduce the above mentioned disadvantages of BOW and improves the prediction abilities in comparison with standard linear kernels by exploiting latent semantics between terms and documents also by increasing the importance of class specific concepts which can be synonymous or closely related in the context of a class.

1.3 Hypothesis

It has been shown that [2], [4], [5],[11], [12], [13]; by replacing the standard kernel functions such as linear kernel with customized kernel functions which take advantage of some background knowledge such as Wikipedia, WordNet it is possible to increase the performance of SVM in the text classification domain. Based on this, we design broad variety of methods in order to explore the capabilities of higher-order paths, class-based meaning values and class-based weighting of terms in both supervised learning and SSLsettings for SVM. Some of the suggested kernels are based on higher-order paths and motivated by the studies of higher-order Naïve Bayes [14], [15] and Higher-Order Smoothing [16], [17] which make use of the higher-order paths between terms, and the work of [18]which focuses on the higher-order paths between both terms and documents. We applied this methodology in a SSL setting with an addition of a new approach to initial labeling of unlabeled data. The other suggested approaches are based on class-based term values. One of them is based on a meaning measure, which calculates the meaningfulness of the terms in the context of classes. The documents vectors are smoothed based on these meaning values of the terms in the context of classes. The other class-based approach is based on the abstract term weights. In these class-based methods, since we efficiently make use of the class information in the smoothing process, they can be considered supervised smoothing kernels.

In our context semantic kernel refers to an approach which extracts or implicitly makes use of semantic relations between terms that seem to be unrelated in Bag-of-words (BOW) model. These semantic relations can be language-wise general relations such as synonyms as explicitly encoded in WordNet or they can be precise to a specific domain such as a particular class of documents.

The first advantage of our suggested solutions is the capability of them to perform much better than standard kernels in terms of classification accuracy. To demonstrate performance improvements, we conduct several experiments on varied benchmark datasets with several different test environments. According to our experimental results our models exceed the performance of linear kernel which is one the state of the art kernels for text classification. In linear kernel, the inner product between two document vectors is used as kernel function, which utilizes the information about shared terms in these two documents. However, our models can take advantage of higher-order paths and class-based values of terms; therefore they extend the boundaries of being a single document as the context.

The second advantage of the proposed approaches is about their simplicity and independency of the outside semantic sources such as WordNet. As a result they can be applied to any language domain without adjustments or parameter optimizations. To show this wide applicability of our kernels we present results with different experimental settings, such as: (i) several datasets from different domains such as newsgroups postings and movie reviews classified into sentiments, (ii) several datasets with different languages such as English and Turkish, (iii) different training portions of the dataset in order to observe the effect of sparsity, and (iv) varying values of misclassification cost (*C*) parameter of the SVM.

The other benefit of our methods is that they also form a foundation that can easily be combined with other term-based semantic similarity methods such as unsupervised semantic similarity measures. It is also possible to combine with similarities between terms derived from a semantic source like WordNet or Wikipedia.

Another advantage of our methods, in particular of our class-based methods is about their execution time. We evaluate this by conducting several experiments. Our class-based methods outperform other corpus-based semantic kernels in many cases in terms of accuracy with less execution time.

Additionally, to the best of our knowledge, all of our kernels are the first studies to use higher-order paths, meaning measure and abstract term weights in a semantic kernel for SVM. We evaluated all of the proposed approaches by conducting a large number of experiments on well-known textual datasets and present results with respect to different experimental conditions. We compare our results with linear kernel which is the traditional kernel used in SVM. Please note that linear kernel is one the state of the art kernels for text classification [19], [20]. Our results show that proposed approaches outperform other kernels in most of the cases.

CHAPTER 2

BACKGROUND AND RELATED WORK

Text classification can be defined as automatically classifying documents according to predefined category-labels, usually by using machine learning algorithms. There are large amounts of textual data accumulated both in organizations and especially on the World Wide Web (WWW) through social networks, blogs, news, forums…etc. This huge set of documents continues to increase by the contribution of millions of people every day. Automatically processing these increasing amounts of textual data is one of the critical problems for research and commercial entities. Text classification is the basis for several important applications such as document filtering and sentiment or opinion classification.

2.1 Traditional Classification Approaches

In machine learning applications there are two conventional strategies; supervised learning and unsupervised learning. Traditional supervised learning algorithms need a set of sufficient labeled data as training set to train the classifier, which will be used to predict the class memberships of the unlabeled instances. On the other hand, unsupervised learning, solely based on unlabeled samples, doesn’t need any labeled data to learn a model. So as to train a classifier, it attempts to discover the indirect structure of unlabeled data [21]. There has been massive amounts of accumulated data on the web, especially on social networks, blogs and forums and continue to increase day by day without any doubt. But unfortunately most of the available data does not have pre-assigned labels which limit their use in several practical machine learning application fields such as text classification, sentiment recognition, speech recognition. Moreover, generally it can be time-consuming, tedious and expensive to assign labels to them manually. Most importantly, learning a classifier using a small number of labeled training data may not generate sufficient performance. In situations where labeled data is inadequate, many algorithms have suggested exploiting and utilizing the unlabeled data to support to learning process for better classification. SSL approaches utilize not only labeled data but also unlabeled data to increase the classification accuracy.

2.1.1 Supervised Learning

Supervised Classification is a supervised task, where supervision is provided in the form of a set of labeled training data, each data point having a class label selected from a fixed set of classes [22]. The goal in supervised classification is to learn a model from the training data that gives the best prediction of the class label of unseen (test) data points.The learned model is represented in the form of classification rules, decision trees, or mathematical formulae. Then, the model is used for classification. First, the predictive accuracy of the model (or classifier) is estimated. A simple way is to use a test set of class-labeled samples, which are randomly selected and are independent of the training samples. The accuracy of a model on a given test set is the percentage of test set samples that are correctly classified by the model. If the accuracy of the model is considered acceptable, the model can be used to classify future data of which the class label is not known[22].

The popular classification techniques are listed below:

- Decision Tree Induction
- Bayesian Classification
- Genetic Algorithms for Classification
- kNearest Neighbor Classifier (k-NN)
- Support Vector Machines (SVMs)

2.1.1.1 Support Vector Machines

Support Vector Machines (SVM) was first proposed by Vapnik, Guyon and Boser [23]. A more detailed analysis is given in [24]. In general, SVM is a linear classifier that aims to finds the optimal separating hyperplane between two classes. The common representation of linearly separable space is

$$w^T \phi(d) + b = 0 \tag{2. 1}$$

where *w* is a weight vector, *b* is a bias and *d* is the document vector to be classified. The problem of finding an optimal separating hyperplane can be solved by linearly constrained quadratic programming which is defined in the following equations:

minimize: $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} \xi_i$ (2.2)

with the constraints

$y_i(w^T \phi(d_i) + b) \geq 1 - \xi_i$, $\xi_i \geq 0$, $\forall_i$ (2.3)

where $\xi = (\xi_1, \xi_2 ... \xi_l)^T$ is the vector of slack variables and C is the regularization parameter, which is used to make a balance between training error and generalization, and has a critical role: if it is chosen as too large, there will be a high penalty for non-separable points, many support vectors will be stored, and the model will overfit; on the other hand if it is chosen too small, there will be underfitting [25].

The problem in Eq. (2.2) can be solved using the Lagrange method [25]. After thesolution the resultant decision function can be formulated as:

$$f(x) = \text{sgn}(\sum_{i=1}^{l} \alpha_i y_i k(d_i, d_j) + b) \tag{2.4}$$

where α_i is a Lagrange multiplier, *k* is a proper kernel function and samples d_i with $\alpha_i > 0$ are called support vectors. An important property of a kernel function is that it has to satisfy Mercer's condition which means being semi-positive[25]. We can consider a kernel function as a kind of similarity function, which calculates the similarity values of data points, documents in our case, in the transformed space. Therefore, defining an appropriate kernel has the direct effect on finding a better representation of these data points as mentioned in in [2], [11]. Popular kernel functions include linear kernel, polynomial kernel and RBF kernel:

- Linear kernel: $k(d_i, d_j) = d_i d_j$ (2.5)
- Polynomial kernel: $k(d_i, d_j) = (d_i d_j + 1)^q, q = 1,2..etc.$ (2.6)
- RBF kernel: $k(d_i, d_j) = \exp(\gamma \|d_i - d_j\|^2)$ (2.7)

For the problems of multiclass classification where there are more than two classes, a decomposition methodology is used to divide it into sub problems. There are basically two categories of multiclass methodology [26]: the all-in-one approach considers the data in one optimization formula [27], whereas the second approach is based on decomposing the original problem into several smaller binary problems, solving them separately and combining their solutions. There are two widely used strategies for this category: “one-against-the-rest” and “one-against-one” approaches [20],[26].It is possible and common to use a kernel function in SVM which can map or transform the data into a higher dimensional feature space if it is impossible or difficult to find a separating hyperplane between classes in the original space; besides SVM can work very well on high dimensional and sparse data [19]. Because of these benefits of SVM, linear kernel is one of the best performing algorithms in text classification domain since textual data representation with BOW approach is indeed quite sparse and requires high dimensionality.

Originally, SVMs were developed to perform binary classification. However,in real world classification problems involve more than twoclasses. A number of methods to generate multiclass SVMs from binary SVMs have beenproposed by researchers and is still a continuing research topic [28].There are two strategies[29]:

The One-Against-All (1AA) approach represents the earliest and most common SVM multiclass approach [30]. This method is also called winner-take-all classification. It assumes that the dataset is tobe classified into *M* classes: Therefore, *M* binary SVM classifiers may be created whereeach classifier is trained to distinguish one class from the remaining *M-1* classes. Forexample, class one binary classifier is designed to discriminate between class one datavectors and the data vectors of the remaining classes. Other SVM classifiers areconstructed in the same manner. During the testing or application phase, data vectors areclassified by finding margin from the linear separating hyperplane. The final output isthe class that corresponds to the SVM with the largest margin.

The One-Against-One (1A1) approachis another methodology in whichSVM binary classifiers for all possible pairs of classes are created[31], [32]. The output from each classifier in the form of a class label is obtained. Theclass label with the highest frequency is assigned to that point in the data vector. In case of a tie,a tie-breaking strategy may be adopted. A common tie-breaking strategy is to randomlyselect one of the class labels that are tied. The main disadvantage of this method is the increase in thenumber of classifiers as the number of class increases. For example, for 7 classes of interest, 21 classifiers will be created.

SVM is powerful to approximate any training data and generalizes good results on given datasets. The complexity in terms of kernel affects the performance on new datasets [34]. SVM supports parameters for controlling the complexity and above all SVM does not tell how to set these parameters and one should be able to determine these parameters by Cross-Validation on the given datasets [22], [33]. The major strengths of SVM are the training is relatively easy. It scales relatively well to high dimensional data and the trade-off between classifier complexity and error can be controlled explicitly.

Choosing the most appropriate kernel and its parameters highly depends on the problem at hand. The choice of a kernel depends on the problem at hand because it depends on what is trying to be modeled. For example, radial basis functions allowpicking out circles (or hyperspheres) - in contrast with the linear kernel, which allows only picking out lines (or hyperplanes).The motivation behind the choice of a particular kernel can be very intuitive and straightforward depending on what kind of information is being expected to extract about the data[37]. Automatic kernel selection is possible and is discussed in the works by Tom Howley and Michael Madden[36].So the weakness of SVM includes choosing the most appropriate kernel function [33], [34], [35], [38].

2.1.1.2 Semantic Supervised Approaches for Text Classification

Linear kernel has a wide usage in the domain of text classification due to the high dimensionality of the representation. As shown in Eq. (2.5) the kernel function is based on the inner products of feature vectors of the documents. So a linear kernel captures similarity between two documents as much as the terms they share. This yields certain problems due to the nature of natural language such as synonymy and polysemy since it is not considering semantic relations between terms. This can be addressed by integrating semantic information between words using semantic kernels such as [2], [4], [8], [11], [27], [39], [40].

According to the definition given in [2], [8], [23], [25],[41] any function in the form given in Eq. (2.8) is a valid kernel function:

$$k(d_p, d_q) = \langle \bar{\phi}(d_p), \bar{\phi}(d_q) \rangle \quad (2.8)$$

whered_p and d_q are input space vectors and $\bar{\phi}$ is a suitable mapping from an input space into a feature space.

As mentioned in Section 1, studies which enhance the representation of documents can be categorized according to their design principles follows:

- Domain knowledge based systems: These systems use ontology or thesaurus to capture the concepts in the documents and incorporate the domain knowledge of separate terms into the words for representation in textual data. They enhance the representation of terms by taking advantages of semantic relatedness among terms. These include [2], [6], [7], [8], [9], [10], [11]. For instance in[11], WordNet [12] is used to calculate semantic similarity between English words. The study in [41] uses super concept declaration with some distance measures between words from WordNet like Wu-Palmer Measure, Inverted Path Length (IPL), Lin Measure and Resnik Measure. A recent study of this kind uses HowNet as a Chinese semantic knowledge-base[13]. In[42], the authors present distinct types of approaches to combine the background knowledge in ontologies into the representation of textual data and show the improvement in the classification accuracy. Similar works also can be found in[10], [43].
- Statistical approaches: These systems use statistical analysis depending on the relations of terms in the set of training documents to expose latent similarities between them[44]. One of the well-known corpus-based systems is Latent Semantics Analysis (LSA) [45]which partially solves the synonymy problem.
- Hybrid methods: These approaches combine the information gathering from ontology and statistical analysis results from the corpus as in[9]. There is a recent survey in [7] about these methods.
- Word sequence enhanced systems: These types of representations treat the words as string sequences. Typically the main idea is based on a word sequence taken out from documents by customary string matching technique. N-gram based representation [46] and similar works in[47], [48], [49]are conventional examples of these kinds of systems.

- Linguistic enriched methods: They make use of syntactic and lexical rules to extract the noun phrases, terminologies and entities from documents and improve the representation using these linguistic units. For instance in [50] multi-words are used to expand the effectiveness of text retrieval system. Also Lewis [51] compares the word-based indexing and phrase-base indexing for representation of documents.

Siolas and d'Alché-Buc in [11] propose a semantic kernel that is intuitively generated from the semantic relations of English words in WordNet. The hierarchies and connections between terms in WordNet are used to calculate semantic relatedness between two words. They benefit from this information to enrich the Gaussian kernel. According to this study, using a semantic similarity metric as a smoothing technique increases the correct classifications.

Bloehdorn et al.[41], uses a semantic kernel with super-concept declaration. Their purpose is to create a kernel function that captures the knowledge of topology that belongs to their super-concept expansion. This kernel function is given in Eq. (2.9), where Q is a semantic smoothing matrix. The Q is composed of P and P^T which contains super-concept information about the corpus. Their results show that they get remarkable improvement in performance, particularly in situations where the feature representations are highly sparse or little training data exists[41].

$$k(d_p, d_q) = d_p P\ P^T d_q^T \qquad (2.9)$$

Bloehdorn and Moschitti [5] designed a Semantic Syntactic Tree Kernel (SSTK) by combining syntactic dependencies like linguistic structures with semantic knowledge that is gathered from WordNet. Similarly, in[8], WordNet is used as a resource of semantic knowledge base. Nevertheless, they express that WordNet's coverage is not sufficient and a more extensive background knowledge resource is required. This is also one of the key reasons that other studies aim to use resources with a wider coverage such as Wikipedia.

One of those works is [1]. The similarity between two documents in the kernel function designed as in Eq. (2.9)and P is a semantic proximity matrix that is created from Wikipedia. The semantic proximity matrix is composed of three measures: 1) content-based similarity measure which depends on Wikipedia articles' BOW representation, 2) out-link category-based measure that brings an information related to the out-link categories of two associative articles in Wikipedia, 3) distance measure which is calculated as the length of the shortest path connecting the two categories of two articles belong to, in the Wikipedia's category taxonomy. The authors state that adding semantic knowledge that is extracted from Wikipedia into document representation overcomes some of the shortages of the BOW approach and improves the accuracy.

The study in [4] also use WordNet to build a semantic proximity matrix based on Omiotis[40], which is a knowledge based measure for computing the relatedness between terms. Nasir et al. incorporated this measure into a (Term Frequency-Inverse Document Frequency) TF-IDF weighting approach. They show that their Omiotis embedded methodology is superior to standard BOW representation. Nasir et al. further broadened their work by taking just top-k semantically related terms and utilizing some evaluation metrics on larger text datasets [9].

Semantic Diffusion Kernel is presented and studied in[27], [39]. Such a kernel is obtained by an exponential transformation on a given kernel matrix as in

$$K(\lambda) = K_0 \exp(\lambda K_0) \qquad (2.10)$$

where K_0 is the gram (kernel) matrix of the corpus in BOW representation and λ is the decay factor. As stated in [27] the kernel matrix K_0 is created by using Eq. (2.11)

$$G = DD^T \qquad (2.11)$$

where *D* is the term by document matrix of the corpus. In [27], [39] it has been demonstrated that $K(\lambda)$ relates to a semantic matrix $\exp(\lambda \frac{G}{2})$ as in Eq. (2.12).

$$S = \exp(\frac{\lambda}{2} G) = \frac{1}{2}(2I + \lambda G + \frac{\lambda^2 G^2}{2!} + \ldots + \frac{\lambda^\theta G^\theta}{0!} + \ldots) \qquad (2.12)$$

where *G* is a generator that displays the initial semantic similarities between words and *S*is the semantic matrix of the exponential of the generator. According to the experiments in [27] the diffusion matrix exploits higher-order co-occurrences to gather latent semantic relationships between terms in the WSD tasks from SensEval.

Zhang et al. [52], [52], focuses on the usage of multi-word phrases for text representation in the task of text classification. In their work they extract multi-word phrases by using the syntactical structure of the noun phrases. They present two strategies which are called general concept representation and subtopic representation, to represent the documents using extracted multi-word phrases[52]. Their first strategy is based on the usage of general concepts for the representation of documents. The second strategy uses the subtopics of the general concepts of representation. Following this they carry out a serious of experiments with SVM linear kernel and non-linear kernels in order to see the effects of using multi-word phrases in a kernel function. They express the benefits of using multi-word phrases with the following aspects[52]: Firstly, using multi-word phrases decreases the number of dimensions. Secondly, acquiring multi-word phrases is an easy task. Thirdly, multi-word phrases carry more semantics than individual words. According to their experimental results, their approach with multi-word linear kernel outperforms the standard linear kernel[52].

2.1.2 Semi-Supervised Learning

There has been massive amounts of accumulated data on the web, especially on social networks, blogs and forums and continue to increase day by day without any doubt. But unfortunately most of the available data does not have pre-assigned labels which limit their use in several practical machine learning application fields such as text classification, sentiment recognition, speech recognition. Moreover, generally it can be time-consuming, tedious and expensive to assign labels to them manually. Most importantly, learning a classifier using a small number of labeled training data may not generate sufficient performance. In situations where labeled data is inadequate, many algorithms have suggested exploiting and utilizing the unlabeled data to support to learning process for better classification. Semi-supervised Learning (SSL) approaches utilize not only labeled data but also unlabeled data to increase the classification accuracy. Recently, SSL has become popular and gained increased attention of both academic and commercial platforms as a new machine learning strategy. SSL is different from two ordinary classification approaches by the usage of unlabeled data to mitigate the effect of insufficient labeled data on classifier accuracy.

Many SSL algorithms have been offered in the past decades, like co-training [53], self-training [54],[55] graph-based methods [56], semi-supervised support vector machines[21], Expectation-Maximization (EM) with generative mixture models [57], transductive support vector machines[58].

Semi-supervised learning can be either transductive or inductive. Most of the learning models and systems in artificial intelligence apply inductive inference where a model is derived from data and this model is further applied on new data. The model is created without taking into account any information about a particular new data vector. The new data would fit into the model to certain degree (an error is estimated). The model is in most cases a global model, covering the whole problem space. Creating a global model that would be valid for the whole problem space is a difficult task and in most cases but; it is not necessary. The output for a new vector is calculated based on the activation of one or several neighboring local models (rules). The inductive learning and inference approach is useful when a global model of the problem is needed even in its very approximate form, when incremental, on-line learning is applied to adjust this model on new data and trace its evolution[59]. A block diagram of an inductive reasoning system is shown in Figure 2.1.

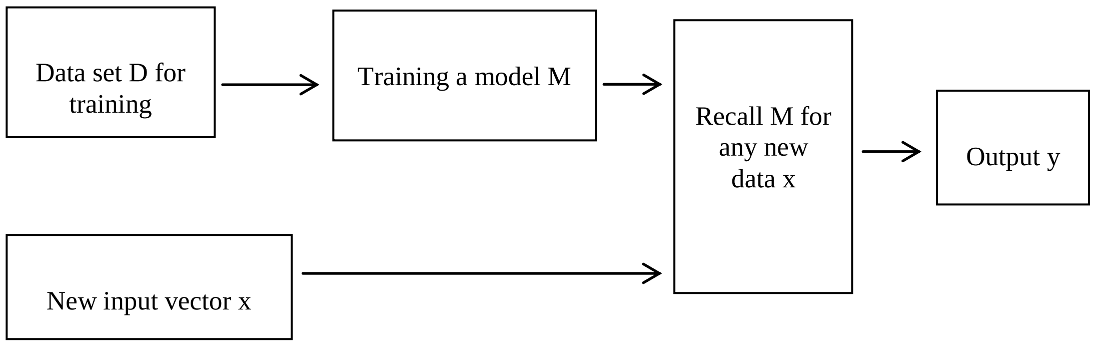


Figure 2.1A block diagram of an inductive reasoning system.

A simple transductive inference method is the *k*-nearest neighbor method (k-NN), where a new data vector x_i is classified into one of the existing classes in the data samples from *D* based on the majority of classes among *k* nearest to the new vector samples that form the set D_i. The distance is measured as Euclidean distance or as another type of distance. In terms of prediction systems, the output value y_i for the new vector x_i is calculated as the average value of the output values of the k-nearest samples from the data set D_i[59]. A block diagram of a transductive reasoning system is shown in Figure 2.2

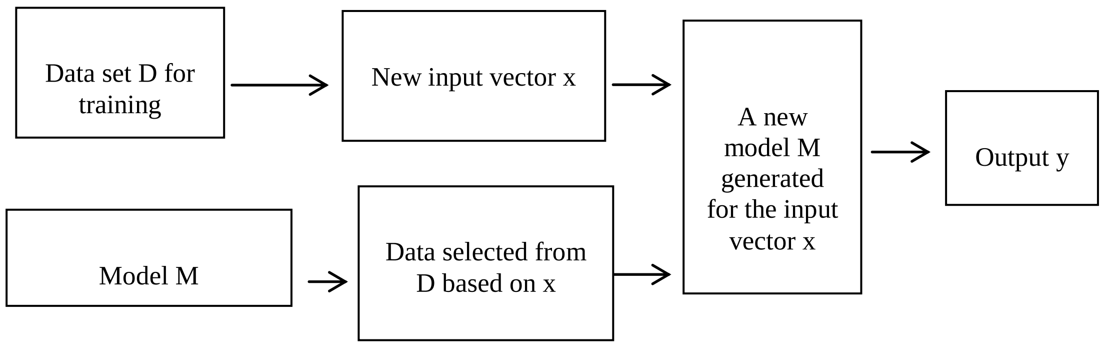

Data set D for training
New input vector x
A new model M generated for the input vector x
Output y
Model M
Data selected from D based on x

Figure 2.2A block diagram of a transductive inference system.

2.1.2.1Semi-Supervised Learning Approaches

Semi-Supervised Learning (SSL) methods make use of unlabeled examples to build better classifiers with higher accuracy when there are a few amounts of labeled training examples. In a typical SSL scenario there are labeled training examples (*L*) and unlabeled examples (*U*). Co-training and Self-training are two popular SSL algorithms. Self-training works as follows [54]; a classifier is constructed from *L* and used to estimate the labels for samples in *U*. Then m unlabeled samples that the classifier has high classification confidence in *U* are assigned labels and moved to extend *L*. After that, the classifier is re-trained using the enlarged data set *L*. Although it is a very simple algorithm; since it is not easy to guarantee the convergence of it, the latter three steps are commonly repeated for a pre-defined maximum iteration number of times or reaching up until some heuristic convergence standard (i.e., there is no remaining unlabeled instances in *U*).

Co-training works like self-training except that it assumes that attributes can be divided into two different views. According to co-training algorithm input features are logically partitioned into two independent groups, and two separate classifiers are trained on these two subsets in labeled data set (*L*) [53]. Following this, each classifier is attempted to label the unlabeled samples in *U*; to put it in more detail; for each classifier, the instances in *U* with the highest classification confidence are selected and added to the labeled data set *L*. Consequently these two classifiers can assist for enlarging the data set *L*. Both classifiers are retrained on this expanded data set, and the steps are re-performed a fixed number of times. Thus each classifier then categorizes the unlabeled data, and teaches the other classifier with some unlabeled instances (those have their newly estimated labels with high classification confidence). Each classifier will be trained again with the supplementary training examples specified by the other classifier, and the procedure is repeated for higher accuracy[21],[53],[60]. As it is discussed in [61] the main idea in co-training is that a classifier may give suitable labels to some samples whereas it may be challenging for the other classifier to do so. Therefore, each classifier can enlarge the training set with instances which are actually informative and important for the other classifier.

Different kinds of self-training and co-training algorithms have offered by researchers over the years. One variant of them is to use entire unlabeled data in each iteration therefore there will no need for the selection criterion. One example of this kind is presented in [62]. The labels of the all unlabeled examples are predicted and then used to extend the training set and update the classifier at all iterations. In [57], co-EM (co-Expectation Maximization) is presented which uses all the unlabeled samples rather than a number of samples chosen from the data pool. Another kind of approaches is to use active learning technique to choose unlabeled examples and then ask some human experts to label them which yields no mislabeled examples will occur, in principle. In [63], a system with active learning is applied to choose unlabeled examples for the multi-view semi-supervised co-EM algorithm. In [64], in each iteration, uncertainty sampling is used to select unlabeled instances, then a cost-sensitive classifier is built on the expanded labeled data and all unlabeled instances with assigned labels. EM algorithm is used in an active learning framework in order to improve SSL in RBF and is applied to content-based image retrieval in [111]. A very similar method (with the addition of suitable preprocessing of the data) is described in [112] for text classification. Nevertheless active learning methods are difficult to apply since they cannot be accomplished without human experts.

Confidence selection is a popular instance selection criterion, which selects unlabeled instances to add to the training set which are classified with high classification confidence [53],[54], [57], [65],[66]. Other selection methods have also been suggested by researchers. For instance, Wang et al., [66], offered an adapted value difference metric as the selection technique in self-training. Their approach is based on decision tree classifiers and used to classify sentences as subjective or objective. They use the Naive Bayes trees algorithm, in order to build a Naive Bayes Classifier at each leaf of the tree. Their approach works well on very small datasets. In [67], a new data editing approach, named SETRED, is presented. Their approach benefits from the information of the neighbors of each self-labeled instance to recognize and remove the mislabeled samples from the self-labeled data. In [68], ISBOLD selection strategy is used to roughly prevent possible performance degradation in self-training and co-training.

Li et al. [69]presents a new methodology which uses three learners. According to [69]L denotes the labeled example set, h_1, h_2 and $h_3$indicate initial learners and U show the unlabeled example set and x is an example in U. Firstly, three classifiers are trained from labeled examples. Then, any two of those classifiers are used to label the unlabeled sample x, if two of them predict the same label; then that example will be utilized to teach the third classifier. It repeats this procedure until none of h_1, h_2 and h_3 changes. The final estimation is accomplished with a majority vote among all the learners.

Ideally, the selected unlabeled examples (together with the assigned labels) can finally assist to learn a better classifier. However, Cozman (2003) [70]stated that unlabeled data may degrade classification accuracy in some extreme situations and when the model assumptions are not correct. For instance in [71], an extensive empirical study was conducted on several popular SSL algorithms (including co-training and self-training) using different base Bayesian classifiers. According to their results on 26 UCI datasets, if the classifier has poor performance and incorrectly assigns labels to some self-labeled examples, there will be accumulated mislabeled data which yields the final performance will be jeopardized. McCallum and Nigam (1998) [62]mention that, they get better classification performance by combining a small set of labeled samples with a large set of unlabeled data with EM . Unfortunately, there are many studies show that unlabeled samples are quite often detrimental to the performance of classifier in many situations [70]. According to those studies the more unlabeled data are joined with a fixed number of labeled instances, the poorer is the classification performance of the corresponding classifier. Therefore, it is obvious that, the classifier should have a good classification performance on the original labeled data if it desires to have good prediction performance on future data. More accurately, utilizing the accuracy on the original labeled data to select more reliable unlabeled samples seems critical for the final classification performance of the SSL algorithm.

In [72], a C4SVM algorithm is presented, which includes misclassification costs into the optimization function of a semi-supervised SVM. In some algorithms only one base learner is applied, which use the unlabeled samples iteratively based on its own knowledge. Some approaches include using EM algorithm to estimate posterior parameters of a generative model, Naive Bayes, by labeling each unlabeled sample, i.e. a probability for each class as it is done in [62]; using the unlabeled data to search for a better configuration of Bayesian Network [73]; using a transductive inference for SVM on a special test set [19]. The self-training algorithm[57] is of that kind, where all iterations the learner converts the most confidently classified unlabeled sample of each class into a labeled training example. These methods and their variants are also described, analyzed and compared in[65]. Furthermore there is a comprehensive survey that includes almost all of well-known semi-supervised learning algorithms in [21].

Transductive Support Vector Machines (TSVMs) is an extension of traditional SVM with the contribution of unlabeled data. The aim is to predict the labels of the unlabeled data and use them in the training step; therefore a linear boundary has the maximum margin on the labeled data (the original labeled data with the addition of labeled unlabeled data this time). One of the recent studies is presented in [113] which is the implementation of semi-supervised support vector machines (S3VMs). Li et al. name their approach as S4VMs and explain that S4VMs tries to exploit multiple candidate low-density separators in contrast to common S3VMs which typically focus on approaching one optimal low density separator. Their comprehensive experiments validate the effectiveness of S4VMs.Also there is a recent study [114]in which several semi-supervised methods and applications are described.

2.2 Higher-Order Co-Occurrence Paths

There are numerous systems with higher-order co-occurrences in text classification. One of the most widespread of them is the Latent Semantic Indexing (LSI) algorithm. The study in [74] verified arithmetically that performance of LSI has a direct relationship with the higher-order paths. LSI’s higher-order paths extract “latent semantics”[15], [74]. Based on these work, the authors in [14], [15] built a new Bayesian classification framework called Higher-Order Naive Bayes (HONB) which presents that words in documents are strongly connected by such higher-order paths and that they can be exploited in order to get better performance for classification. Both HONB [14] and HOS [16], [17] are based on Naïve Bayes.

A higher-order path can also be represented as a chain of co-occurrences of entities (attribute values, words, terms, etc.) in different records (instances, documents, etc.). Actually they can extract co-occurrence relations from virtually any dataset as long as there is a meaningful context of entities[15]. Kontostathis et al.[74] proved mathematically and demonstrated empirically that LSI is based on the use of higher-order relations, in particular higher-order co-occurrences. The authors also demonstrated that the retrieval performance of LSI is correlated with higher-order relations. Higher-order relations in LSI capture “latent semantics” [15], [74].

Ganiz et al. claimed that their results on several textual datasets show that when training data is scarce (i.e., a small number of labeled instances), HONB significantly reduces the generalization error by leveraging higher-order paths[15].

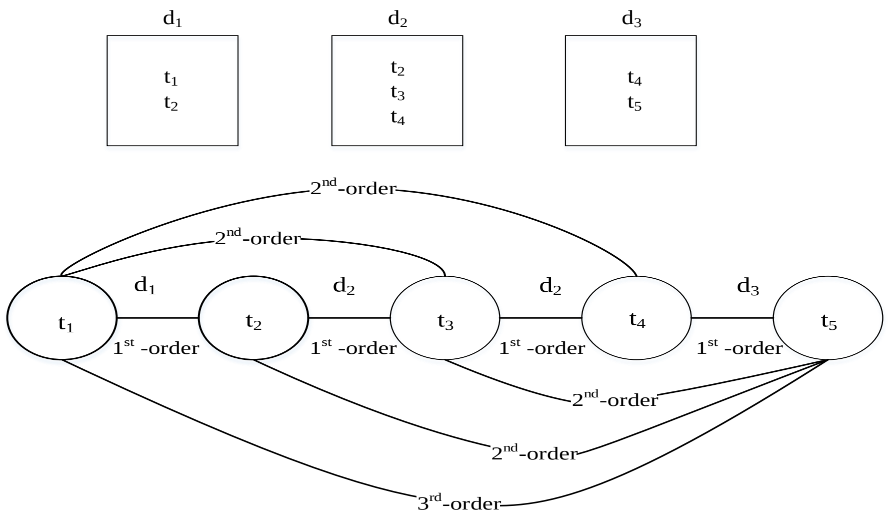
Benefits of using on higher-order paths between documents between terms[14], [17] are demonstrated inFigure 2.3. There are three documents, d_1, d_2, and d_3, which consist of a set of terms $\{t_1, t_2\}$, $\{t_2, t_3, t_4\}$,and $\{t_4, t_5\}$, respectively. Using a traditional similarity measure which is based on the common terms (e.g. dot product), the similarity value between documents d_1 and d_3 will be zero since they do share any terms. But this measure is misleading since these two documents have some connections in the context of the dataset over d_2 as it can be perceived inFigure 2.3. This supports the idea that using higher-order paths between documents, it is possible to obtain a non-zero similarity value between d_1 and d_3 which is not possible in the BOW representation. This value turns out to be larger if there are many interconnecting documents like $d_2$between d_1 and d_3. This is caused by the fact that the two documents are written on the same topic using different but semantically closer sets of terms.

InFigure 2.3, there is also a higher-order path between t_1 and t_3. This is an illustration of a novel second-order relation since these two terms do not co-occur in any of these documents and can remain undetected in traditional BOW models. However, we know that t_1 co-occurs with t_2 in document d_1, and t_2 co-occurs with t_3 in document d_2. The same principle that is mentioned in the case of documents above applies in here. The similarity between t_1 and t_3 becomes more eminent if there are many interconnecting terms such as t_2 or t_4 and interconnecting documents like d_2. The regularity of these second-order paths may reveal latent semantic relationships such as synonymy [17].

2.3 Term Weighting Methods

There are different approaches to assign appropriate weights to the terms to improve the classification performance: For example; binary, TF, TF-IDF [75], [76] and its variants are the traditional methods borrowed from IR field and belong to the unsupervised term weighting methods. Also there are approaches which have proper place in the supervised term weighting category since term weights are calculated according to the category membership information of training documents. One type of them is to weight terms by

using feature selection metrics, i.e. gain ratio, Information Gain (IG), odds ratio and so



1st-order term co-occurrence {t_1, t_2}, {t_2, t_3}, {t_3, t_4}, {t_2, t_4}, {t_4, t_5}
2nd-order term co-occurrence {t_1, t_3}, {t_1, t_4}, {t_2, t_5}, {t_3, t_5}
3rd-order term co-occurrence {t_1, t_5}

Figure 2.3Graphical demonstration of first-order, second-order and third-order paths between terms through documents[93].

on, in[77], [78]. Another recent approach that improves the terms' discriminating power for text categorization task is Term Frequency-Relative Frequency (TF-RF)[79]which considers only the frequency of relevant documents (i.e. those which contain this term). Furthermore [80], [81] is inspired from Term Frequency-Inverse Class Frequency (TF-ICF)[82], [83] and extends the boundaries of traditional weighting method TF-IDF [75] by the contribution of category information of terms in the training set.

TF-IDF [75] is the most popular term weighting method. Its formula is given in Eq. (2.14), where tf_w represents the frequency of the term w in the document and IDF is the inverse of the document frequency of the term in the dataset. IDF's formula is also given in Eq. (2.13) where $|D|$ denotes the total number of documents and df_wrepresents the number of documents which contains term w. TF indicates the occurrence of word w in document d_i. The TF-IDF has proved extraordinarily robust and difficult to beat, even by much more carefully worked out models and theories [84].

$$IDF(w) = \frac{|D|}{df_w} \qquad (2.13)$$

$$TF - IDF(w, d_i) = tf_w \times \log(IDF(w)) \qquad (2.14)$$

A similar but supervised version of TF-IDF is called TF-ICF whose formula given in Eq. (16) as in[82], [83]. In Eq. (2.16), $|C|$ represents number of classes and cf_w indicates the number of classes which contains term w.

$$ICF(w) = \frac{|C|}{cf_w} \tag{2.15}$$

$$TF - ICF\left(w, c_j\right) = \sum_{d \in c_j} tf_w \times \log\left(ICF(w)\right) \tag{2.16}$$

In [79]a new term weighting method is proposed with the idea of simplifying a multi-label classification problem into multiple independent binary classification problems. In their methodology, a chosen category is tagged as the positive category and all the remaining categories in the same dataset are combined together as the negative category. According to their methodology; the more focused a high frequency term is in the positive category than in the negative category, the more contributions it makes in selecting the positive samples from the negative samples. Their term weightings formula is as follows:

$$TF - RF = tf_w \times \log\left(2 + \frac{a}{\max(1, c)}\right) \tag{2.17}$$

where $tf_w$is the term frequency of word w, a is the number of documents in the positive category which contain term w and c is the number of documents in the negative category which contain term w. Table 2.1 demonstrates the difference between the discriminative powers of both IDF and RF. Table 2.1 lists the IDF and RF values of four terms based on two categories, namely, 00_acq and 03_earn respectively. The first two terms, acquire and stake, are closely related to the theme discussed in category 00_acq while the last two terms, payout and dividend, are closely related to the theme discussed in category 03_earn. However, the IDF disregards the category or label information of the training set. Thus, each of these four terms is weighted equally by the IDF even in terms of the two different categories. On the other hand, by using the RF scheme which pays attention to category information, each term is assigned more appropriate weights in terms of different categories[79].

By being inspired from the idea of both IDF and ICF, a new term weighting method which is designed as a part of a feature extraction algorithm is proposed in [80], [81]. According to their approach the effect of a term over a class is calculated as follows:

$$W_{w,c} = \log(tfc_{w,c} + 1) \times \log(\frac{N}{N_w}) \quad (2.18)$$

where *tfc* $_{w,c}$ is the total term frequency of word *w* in the documents of class *c*, *N* is the total number of documents in the corpus and N_w is the total number of documents those contain term *w*. By using this class-dependent term weighting scheme they develop a feature extraction method for text classification. They carry out experiments on benchmark datasets to compare the classification performance with well-known feature extraction algorithms. Their experimental results show that using this feature extraction with a class-dependent term weighting scheme enhances classification performance on the classifiers they use when compared with other feature extraction methods.

Table 2.1Comparison of the weights of four features in Category 00_acq and 03_earn ([79])

Feature	**Category:00_acq**		**Category:03_earn**	
	IDF	**RF**	**IDF**	**RF**
acquire	3.553	4.368	3.553	1.074
stake	4.201	2.975	4.201	1.082
payout	4.999	1	4.999	7.820
dividend	3.567	1.033	3.567	4.408

2.4 Helmholtz Principle from Gestalt Theory and Meaning Calculation

According to Helmholtz principle from Gestalt theory in image processing; “observed geometric structure is perceptually meaningful if it has a very low probability to appear in noise” [100]. This means that events that have a large deviation from randomness or noise can be noticed easily by humans. This can be illustrated inFigure 2.4. In the left hand side of Figure 2.4, there is a group of five aligned dots but it is not easy to notice it due to the high noise. Because of the high noise, i.e. large number of randomly placed dots, the alignment probability of five dots increases.On the other hand, if we remove the number of randomly placed dots considerably, we can immediately perceive the alignment pattern in the right hand side image since it is very unlikely to happen by chance. This phenomenon means that unusual and rapid changes will not happen by chance and they can be immediately perceived.

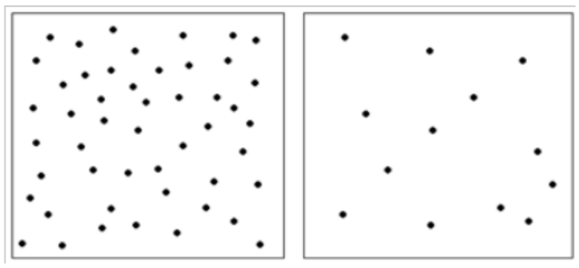


Figure 2.4The Helmholtz principle in human perception (adopted from [100])

As an example, assume you have unbiased coin and it is tossed 100 times. Any 100-sequence of heads and tails can be generated with probability of (½)100 and Figure 2.5is generated where 1 represents heads and 0 represents tails [99].

$$s_1 = 10101\,11010\,01001 \ldots 00111\,01000\,10010$$
$$s_2 = \underbrace{111111111 \ldots 111111}_{50\ times}\ \underbrace{000000000 \ldots 000000}_{50\ times}$$

Figure 2.5The Helmholtz principle in human perception (adopted from [99])

First sequence, s_1 is expectable for unbiased coin but second output, s_2 is highly unexpected. This can be explained by using methods from statistical physics where we observe macro parameters but we don't know the particular configuration. For instance expectation calculations can be used for this purpose[99].

A third example is known as birthday paradox in literature. There are 30 students in a class and we would like to calculate the probability of two students having the same birthday and how likely or interesting is this. Firstly, we assume that birthdays are independent and uniformly distributed over the 365 days of a year. Probability P_1 of all students having different birthday in the class is calculated in Eq. (2.19)[109].

$$P_1 = \frac{365x364x..x336}{365^{30}} \approx 0.294 \qquad (2.19)$$

The probability P_2 of at least two students born on same day is calculated in Eq. (2.20).This means that approximately 70% of the students can have the same birthday with another student in the class of 30 students.

$$P_2 = 1 - 0.294 = 0.706 \qquad (2.20)$$

When probability calculations are not computable, we can compute expectations. The expectation of number of 2-tuples of students in a class of 30 is calculated as in Eq. (2.21). This means that on the average, 1.192 pairs of students have the same birthday in the class of 30 students and therefore it is not unexpected. However the expectation values for 3 and 4 students having the same birthday, E(C3)≈0.03047 and E (C4) ≈0.00056, which are much smaller than one, indicates that these events will be unexpected[109].

$$E(C_2) = \frac{1}{365^{2-1}} \binom{30}{2} = \frac{1}{365} \frac{30!}{(30-2)!\ 2!} = \frac{30x29}{2x365} \approx 1.192 \qquad (2.21)$$

In summary, the above principles indicate that meaningful features and interesting events appears in large deviations from randomness. Meaningfulness calculations basically correspond to calculations of expectations and they stem from the methods in statistical physics[100].

In the context of text mining, the textual data consist of natural structures in the form of sentences, paragraphs, documents, and topics. In[100], the authors attempt to define meaningfulness of these natural structures using the human perceptual model of Helmholtz principle from Gestalt Theory. Modelling the meaningfulness of these structures is established by assigning a meaning score to each word or term. Their new approach to meaningful keyword extraction is based on two principles. The first one states that these keywords which are representative of topics in a data stream or corpus of documents should be defined not only in the document context but also the context of other documents. This is similar to the TF-IDF approach. The second one states that topics are signaled by “unusual activity”, a new topic can be detected by a sharp rise in the frequencies of certain terms or words. They state that sharp increase in frequencies can be used in rapid change detection. In order to detect the change of a topic or occurrence of new topics in a stream of documents, we can look for bursts on the frequencies of words. A burst can be defined as a period of increased and unusual activities or rapid changes in an event. A formal approach to model “bursts” in document streams is presented in[110]. The main intuition in this work is that the appearance of a new topic in a document stream is signaled by a “burst of activity" with certain features rising sharply in frequency as the new topic appears.

Based on the theories given above, new methods are developed for several related application areas including unusual behavior detection and information extraction from small documents [105], for text summarization[101], defining relations between sentences using social network analysis and properties of small world phenomenon [102] and rapid change detection in data streams and documents [99]and also for keyword extraction and rapid change detection[100]. These approaches make use of the fact that meaningful features and interesting events come into view if their deviations from randomness are very large.

The motivating question in these studies is “if the word w appears m times in some documents is this an expected or unexpected event?” [100]. Given that S_w is the set of all words in N documents and a particular word w appears K times in these documents. Then random variable C_m counts m-tuple of the elements of S_w appears in the same document. Following this the expected value of C_m is calculated under the assumption that the words are independently distributed among the documents. C_m is calculated using random variable $X_{i1,i2\ldots im}$ which indicates if words $w_{i1},\ldots,w_{im}$ co-occurs in the same document or not. Based on this the expected value $E(C_m)$ can be calculated as in Eq. (2.23) by summing the expected values of all these random variables for all the words in the corpus.

$$C_m = \sum_{1 \le i1 < \ldots < im \le K} X_{i1,\ldots,im} \qquad (2.22)$$

$$E(C_m) = \sum_{1 \le i1 < \ldots < im \le K} E(X_{i1,\ldots,im}) \qquad (2.23)$$

The random variable $X_{i1,i2\ldots im}$can only take values one and zero. As a result the expectation of this random variable which shows if these m words co-occurs in the same document can be calculated in Eq. (2.24), where N is the total number of documents. “If in some documents the word w appears m times and $E(Cm)<1$ then it is an unexpected event” [100].

$$E(X_{i1,\ldots,im}) = \frac{1}{N^{m-1}} \qquad (2.24)$$

As a result $E(C_m)$ can simply be expressed as in Eq. (2.25) and this expectation actually corresponds to Number Of False Alarms (NFA) of m-tuple of word *w* which is given in Eq. (2.26). This corresponds to the number of times m-tuple of the word *w* occurs by chance[100]. Based on this, in order to calculate the meaning of a word w which occurs m times in a context (document, paragraph, sentence), we can look its NFA value. If the NFA (expected number) is less than one, then the occurrence of m times can be considered as a meaningful event because it is not expected by our calculations but it is already happened. Therefore, word w can be considered as a meaningful or important word in the given context.

$$E(C_m) = \binom{K}{m} \frac{1}{N^{m-1}} \tag{2.25}$$

Based on the NFA, the meaning score of words are calculated using Eq. (2.26) and Eq. (2.27) in[102]:

$$NFA(w,P,D) = \binom{K}{m} \frac{1}{N^{m-1}} \tag{2.26}$$

$$Meaning(w,P,D) = -\frac{1}{m} \log NFA(w,P,D) \tag{2.27}$$

where *w* represents a word, *P* represents a part of the document such as a sentence or a paragraph, and *D* represents the whole document. Additionally, *m* indicates the appearance number of word *w* in *P* and *K* shows the appearance number of word w in *D*. *N*= *L / B* in which *L* is the length of *D* and *B* is the length of *P* in words[102]. To define Meaning function, the logarithmic value of NFA is used based on the observation that NFA values can be exponentially large or small [100].

As mentioned above, the meaning calculations are performed in a supervised setting. In other words, we use a class of documents as our basic unit or context in order to calculate meaning scores for words. In this approach meaning calculations basically show how high a particular words' frequency is expected to be in a class of documents compare to the other classes of documents. If it is unexpected then meaning calculations result in a high meaning score. In this aspect it is similar to the Multinomial Naïve Bayes in which the all the documents in a class are merged into a single document and then the probabilities are estimated from this one large class document. It also bears similarities to TF-ICF approach in which the term frequencies are normalized using the class frequencies.

In supervised meaning calculations, which are given in Eq. (2.31) and Eq. (2.32), parameter $c_j$represents documents which belong to class *j* and *S* represents the complete training set. Assume that a feature *w* appears *k* times in the dataset *S*, and *m* times in the documents of class c_j. The length of dataset (i.e. training set) *S* and class c_j measured by the total term frequencies is *L* and Brespectively. *N* is the ratio of the length of the dataset and the class, which is calculated in Eq. (2.30). The NFAis defined in Eq. (2.31).

$$L = \sum_{d \in S} \sum_{w \in d} tf_w \tag{2.28}$$

$$B = \sum_{d \in c_j} \sum_{w \in d} tf_w \tag{2.29}$$

$$N = \frac{L}{B} \tag{2.30}$$

$$NFA(w, c_j, S) = \binom{k}{m} \frac{1}{N^{m-1}} \tag{2.31}$$

Based on NFA, the meaning score of the word *w* in a class c_j is defined as:

$$meaning(w, c_j) = -\frac{1}{m} \log NFA(w, c_j, S) \tag{2.32}$$

This formula can be re-written as:

$$meaning(w, c_j) = -\frac{1}{m} \log \binom{k}{m} - \left[(m-1) \log N\right] \tag{2.33}$$

The larger the meaning score of a word *w* in a class c_j, the more meaningful, significant or informative that word is for that class.

CHAPTER 3

EXPERIMENTAL SETUP

We integrated our kernel functions into the implementation of the SVM algorithm in WEKA [85]. In other words, we built numerous kernel functions those can be directly used with Platt's Sequential Minimal Optimization (SMO) classifier [86].

In order to see the performance of our proposed algorithms on text classification, we performed a series of experiments on several textual datasets which are shown inTable 3.1. Our first dataset IMDB[1] is a collection of movie reviews. It contains 2,000 reviews about several movies in IMDB. There are two types of labels; positive and negative. The labels are balanced in both training and test sets that we used in our experiments. 1150 Haber is our second dataset. It contains 1150 news-articles within five categories under the titles of magazine, politics, sport, economy and health collected from Turkish online newspapers [87]. Our third dataset is five-class version of the WEBKB[88] dataset, namely WEBKB5, which contains web pages, gathered from different universities' computer science departments. WEBKB5 dataset has highly skewed class distribution. Other datasets are variants of popular 20 Newsgroups[2] dataset. This data set is a collection of approximately 20,000 newsgroup documents, partitioned evenly across 20 different newsgroups and commonly used in machine learning applications, especially for text classification and text clustering. We used four basic subgroups namely,"Politics", "Comp","Science" and "Religion" from the 20 Newsgroups dataset. The documents are evenly distributed to the classes.

[1]http://www.imdb.com/interfaces

[2]http://www.cs.cmu.edu/~textlearning

The sixth dataset we use isthe Mini-newsgroups[1] dataset which has 20 classes and also has a balanced class distribution. This is a subset of the 20 Newsgroups dataset, too. Properties of these datasets are given inTable 3.1.

We apply stemming and stopword filtering to these datasets. Additionally, we filter rare terms which occur in less than three documents. We also apply attribute selection and select the most informative 2,000 terms using IG as described in [14], [15], [16], [17]. This preprocessing increase the performance of the classifier models by reducing the noise. We perform this preprocessing equally in all experiments.

Table 3.1Comparison ofproperties of datasets before attribute selection

Dataset	**#classes**	**#instances**	**#features**
IMDB	2	2,000	16,679
1150 Haber	5	1150	7,948
WEBKB5	5	4,336	12,841
20 Newsgroups-Politics	3	1,500	9,864
20 Newsgroups-Science	4	2,000	9,615
20 Newsgroups-Religion	4	1,500	7,790
20 Newsgroups-Comp	5	2,500	12,151
Mini-newsgroups	20	2,000	12,112

In order to observe the behavior of our semantic kernel under different training set size conditions, we use the following percentage values for training set size: 5%, 10%, 30%, 50%, 70%, 80% and 90%. Remaining documents are used for testing. This is essential since we expect that the advantage of using semantic kernels should be more observable when there is inadequate labeled data.

One of the main parameters of SMO [89] algorithm is the misclassification cost (*C*) parameter. We conducted a series of optimization experiments on all of our datasets with the values of {10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}}. For all the training set percentages we selected the best performing one. The optimized *C* values for the corresponding methods on each dataset at different training levels are given in Table 3.2, Table 3.3, Table 3.4 and Table 3.5. This is interesting because the values vary a lot among datasets and training set percentages (TS).

[1]http://archive.ics.uci.edu/ml/

Table 3.2Optimized C values for IHOSKon differentdatasets

TS %	**20 Newsgroups-Science**	**20 Newsgroups-Politics**	**WEBKB5**	**Mini-Newsgroups**
5	1	10^{-1}	1	1
10	1	10^{-1}	1	1
30	1	10^{-1}	1	10^{2}
50	1	10^{-1}	1	10^{2}
70	1	1	1	10^{2}
80	1	1	1	10^{2}
90	1	10^{-1}	1	10^{1}

Table 3.3Optimized C values for HOTK on different datasets

TS %	**20 Newsgroups-Science**	**20 Newsgroups-Politics**	**20 Newsgroups-Comp**	**Mini-Newsgroups**
5	1	10^{-1}	1	1
10	1	10^{-1}	1	1
30	1	10^{-1}	1	10^{2}
50	1	10^{-1}	1	10^{2}
70	1	1	1	10^{2}
80	1	1	1	10^{2}
90	1	10^{-1}	1	10^{1}

Table 3.4Optimized C values for CMK on different datasets

TS%	**IMDB**	**20 Newsgroups-Science**	**20 Newsgroups-Politics**	**20 Newsgroups-Religion**	**20 Newsgroups-Comp**	**Mini-newsgroups**
5	10^{-2}	10^{-2}	10^{-2}	10^{-2}	10^{-1}	10^{1}
10	10^{-2}	10^{-2}	10^{-2}	10^{-2}	10^{-1}	10^{2}
30	10^{-1}	10^{-2}	10^{-2}	10^{-2}	1	10^{1}
50	10^{-2}	10^{-1}	10^{-2}	10^{-1}	10^{2}	1
70	10^{1}	10^{-1}	10^{-1}	10^{-2}	10^{-1}	1
80	10^{-2}	10^{-2}	10^{-2}	10^{-1}	10^{-1}	1
90	10^{-2}	10^{-2}	10^{-2}	10^{-2}	10^{2}	1

Table 3.5Optimized C values for linear kernel on different datasets

TS%	**IMDB**	**20 Newsgroups-Science**	**20 Newsgroups-Politics**	**20 Newsgroups-Religion**	**20 Newsgroups-Comp**	**Mini-newsgroups**
5	10^{-1}	10^{-1}	10^{-1}	10^{-1}	10^{-1}	1
10	10^{1}	1	10^{-1}	10^{-1}	10^{-1}	1
30	10^{-1}	1	10^{-1}	10^{-1}	10^{-1}	1
50	10^{-1}	1	10^{-1}	1	10^{-1}	10^{1}
70	10^{-1}	1	10^{-1}	10^{-1}	1	10^{1}
80	10^{-1}	1	10^{-1}	10^{-1}	1	10^{1}
90	10^{-1}	1	1	10^{-1}	1	10^{1}

The main evaluation metric in our experiments is accuracy result and in the results tables they are written with their standard deviations. Also Student's t-Tests for statistical significance tests are provided. We use $\alpha = 0.05$ significance level which is a commonly used level. In addition for the accuracy we used the following performance gain equation;

$$Gain_y = (P_y - P_x) / P_x \qquad (3.1)$$

where P_y is the accuracy of SMO with our semantic kernel and P_x stands for the accuracy result of the linear kernel. The experimental results tables include training set percentage (TS), the accuracy results of Linear Kernel, and our other semantic kernels. Also the last columns demonstrate the (%) gain of our proposed kernel over linear kernel calculated as in Eq. (3.1). We run our experiments using our experiment framework called Turkuaz, which closely uses WEKA library.

CHAPTER 4

HIGHER-ORDER SEMANTIC KERNELS

4.1 Higher-Order Semantic Kernel (HOSK)

We propose a novel semantic kernel called HOSK[97] for SVM applied to document categorization that takes advantages of latent semantics in higher-order paths between documents. Although the previous works on higher-order relations focus on the higher-order paths between terms [15], [16]we focus on the higher-order paths between documents. We can create a graph structure where the nodes represents documents and the edges are represents the similarity between documents. One of the most straightforward ways of defining the similarity between two documents is using statistics of shared words. As a result edges between documents in our graph structure are weighted according to the frequencies of the shared words.

The advantage of using shared terms between documents as edges in this graph structure is that it is easier to grasp semantics of similarity between documents. On the other hand, when we go second-order and higher-order paths between documents, it is more difficult to grasp semantic relations compared to the first-order and higher-order paths between terms. These second-order paths may reveal latent similarity caused by synonymous or highly related terms. With the help of higher-order paths, in particular second-order paths, HOSK takes the advantage of getting latent semantics between documents. In this study[97] we extract and use first-order and second-order paths. Especially the second-order paths reveal latent semantics. Let's consider the following example; in Table 4.1 there are no common terms shared by documents d_1 and d_2; this means that by using a classical similarity measure such as the Cosine measure or the Euclidean distance, these documents' similarity is calculated to zero. However, we can see that both d_1 and d_2 share terms with d_3 meaning that words t_2 and t_3 have some similarities in the *document space*. With a higher-order-path approach, it is possible to get a similarity between d_1 and $d_2$which is bigger than zero. We can explain this situation with the possibility that two documents are written on the same topic using two different but semantically closer sets of terms. In this case terms belonging to each set frequently co-occur in other documents relating to this topic, forming a connection pattern which can be revealed by using second-order paths.

Table 4.1A document by term matrix representation of three documents

M	**t_1**	**t_2**	**t_3**	**t_4**
d_1	1	1	0	0
d_2	0	0	1	1
d_3	0	1	1	0

4.1.1Methodology

In our system, matrix D is built from the whole corpus as a classical document by term frequency matrix. Let Dbe the data matrix having r rows (documents) and $c$columns (words) based on the whole corpus; m_{ij} shows occurrence frequency of the j^{th}word in thei^{th}document;

m_i= [m_{i1} .. m_{ic}] is the row vector representing the document i and m^j= [m_{1j} ..m_{rj}] the column vector corresponding to word$j$.

Since we deal with textual datasets with high dimensionality and sparsity, a proper normalization on this initial data matrix is beneficial. We tried many matrix normalization techniques including row-level normalization (dividing each value in a row by the maximum value in that row), column-level normalization (dividing each value in a column by the maximum value in that column), document-length normalization (dividing each term frequency in a row with the corresponding document's length) and other techniques from the literature which are detailed in [90] such as z-score normalization, min-max normalization, etc. We obtain best accuracy results with row-level normalization which is defined below:

$$\forall_i \in 1..r \quad D_i = \frac{m_i}{\max(m_i)} \qquad (4.1)$$

where *r* is the number of documents in the corpus,*m* is the row vector representing the document and D_i is the normalized form of the row vector of *m*.

We then calculate first-order paths matrix namely *F* between documents like in the Eq. (4.2):

$$F = D\,D^T \tag{4.2}$$

In other words, *F* matrix is calculated by multiplying document by term matrix with its transpose. Each value in the matrix of *F* shows the similarity between corresponding documents. For instance the similarity between document_i(m_i) and document_j (m_j) is calculated as follows:

$$F(i, j) = m_{i1} \times m_{j1} + m_{i2} \times m_{j2} + m_{i3} \times m_{j3} + \ldots + m_{ic} \times m_{jc} \tag{4.3}$$

where *c* is the number of terms in the corpus.

F is a (document by document) square matrix whose dimension is the same as the number of documents in the corpus. We observe that *F* has many zero values. Two documents have a non-zero similarity value in *F* only if these two documents share same words. In order to capture the latent semantic information between documents we calculated the second-order paths matrix namely *S* between documents as in the Eq. (4.4):

$$S = F\,F \tag{4.4}$$

By multiplying *F* by itself *S* matrix is formed. Again *S* is a document by document square matrix whose dimension is the same as the number of documents in the corpus. This matrix shows the second-order paths between documents.

Each value in the matrix of *S* shows the similarity between corresponding documents. For instance the similarity between document_i (m_i) and document_j (m_j) is calculated as in Eq. (4.5):

$$S(i, j) = F(m_i, m_1) \times F(m_j, m_1) + F(m_i, m_2) \times F(m_j, m_2) + F(m_i, m_3) \times F(m_j, m_3) + F(m_i, m_4) \times F(m_j, m_4) + \ldots + F(m_i, m_r) \times F(m_j, m_r) \tag{4.5}$$

So, both *F* and *S* have similarity information between documents. We observed that the values in *S* are extremely larger than the ones in *F*. This could be explained with the fact that *S* has values not only based on only common terms between documents but also some indirectly latent semantics. So, we normalize the two matrix separately using Eq. (4.6) and Eq. (4.7):

$$\forall i, j \in 1 \ldots r \quad FN_{ij} = \frac{F_{ij}}{\max(F)} \qquad (4.6)$$

$$\forall i, j \in 1 \ldots r \quad SN_{ij} = \frac{S_{ij}}{\max(S)} \qquad (4.7)$$

where *r* is the number of documents in our corpus, *F* is the matrix shows the number of first-order paths between documents, *FN* is normalized first-order paths matrix, *S* is second-order paths matrix, and *SN* is normalized second-order paths matrix, respectively. After this normalization we combine these two matrices with a weight value λ in order to see the effect of *FN* and *SN* matrices to the accuracy in our experiments. In order to optimize λ, the following values are taken into consideration: 0, 0.25, 0.5, 0.75, 0.8, 0.85, 0.9, 0.95, and 1. Combination of *FN* and *SN* matrices yields final similarity matrix which is shown in Eq. (4.8):

$$\mathrm{Sim}(d_i,d_j) = (\lambda \times \mathrm{SN}_{ij}) + ((1 - \lambda) \times \mathrm{FN}_{ij}) \qquad (4.8)$$

Based on the results we tune the *λ* parameter to the value of 0.95.This is a satisfactory result for us from the aspect of the more contribution of higher-order paths means the more accurate results.

After that, we use this similarity matrix as a kernel (gram) matrix in SVM by plugging in the SMO WEKA’s implementation[85]. In other words we built such a kernel matrix that is directly applicable in Platt’s SMO learner. One of the most important parameters of SMO algorithm is misclassification-cost (*C*) parameter. After a set of optimization experiments we did not observe a significant difference and that’s why we tuned it to its default value of 1.In all of the experiments including not only HOSK but also linear kernel we used this same value.

4.1.2 Experimental Results and Discussion

According to our experiments HOSK demonstrates a notable performance on 1150 Haber dataset, which can be seen in Table 4.2. HOSK outperforms our baseline kernel (Linear Kernel, which is one of the state-of-the-art kernels in text classification) by extensive boundaries in all training set percentages. The performance gain is specifically obvious at low training set levels. For instance at training levels 1%, 5%, and 10% HOSK statistically significantly outperforms Linear Kernel with the gains of 20.39% ,15.82%, 8.81% on Linear Kernel ,respectively.

Table 4.2Accuracy of HOSK and linear kernel on 1150 Haber dataset with varying training set size

TS %	**Linear**	**HOSK**	**Gain**
1	46.99±4.54	**56.57±12.22**	20.39
5	72.82±4.68	**84.34±2.33**	15.82
10	80.51±2.68	**87.60±1.26**	8.81
30	88.55±1.34	**90.78±0.58**	2.52
50	89.72±1.13	**90.90±0.82**	1.32
70	91.59±1.06	**92.41±0.54**	0.90
80	92.30±2.89	**92.43±3.15**	0.14
90	91.83±3.18	**92.17±2.21**	0.37

According to Table 4.3, on WEBKB5 dataset, at training levels 1%, 5%, and 10% HOSK gives statistically significant results over Linear Kernel besides the results that HOSK outperforms than Linear Kernel in all of the training levels.

The same is valid for 20 Newsgroups-Comp and 20 Newsgroups-Science datasets, where HOSK outperforms Linear Kernel in all training levels. This can be seen from Table 4.4 and Table 4.5. The performance improvement is most visible in small training set levels for instance in train split 1%, HOSK can achieve an accuracy of 72.59% where the Linear Kernel accuracy is only 52.16% for 20 Newsgroups-Science dataset, which can be clearly seen from Table 4.5.

At small training data levels first-order methods give zero as the similarity of two instances those do not contain common words. But by the use of higher-order paths the similarity between those two instances can be larger than zero.

Table 4.3Accuracy of HOSK and linear kernel on WEBKB5 dataset with varying training set size

TS %	**Linear**	**HOSK**	**Gain**
1	59.74±3.48	**75.1±2.82**	25.71
5	72.77±1.43	**84.20±0.87**	15.71
10	78.46±1.60	**86.79±0.62**	10.62
30	84.88±0.97	**88.45±0.46**	4.21
50	86.40±0.96	**89.11±0.48**	3.14
70	88.05±1.05	**89.61±0.65**	1.77
80	87.77±1.61	**89.68±1.01**	2.18
90	88.28±1.18	**89.15±1.56**	0.99

Table 4.4Accuracy of HOSK and linear kernel on 20 Newsgroups-Compdataset with varying training set size

TS %	**Linear**	**HOSK**	**Gain**
1	34.93±4.31	**49.43±2.80**	41.51
5	53.73±4.47	**65.48±2.56**	21.87
10	62.28±2.57	**70.66±1.07**	13.46
30	73.97±1.62	**74.69±1.09**	0.97
50	77.87±1.60	**78.54±1.15**	0.86
70	79.57±1.74	**81.11±1.66**	1.94
80	78.88±2.31	**80.92±1.98**	2.59
90	80.88±2.60	**83.04±2.48**	2.67

Table 4.5Accuracy of HOSK and linear kernel on 20 Newsgroups-Science dataset with varying training set size

TS %	**Linear**	**HOSK**	**Gain**
1	52.16±5.25	**72.59±5.84**	39.17
5	70.93±3.89	**85.69±1.80**	20.81
10	77.74±3.52	**87.87±1.34**	13.03
30	86.73±1.32	**93.11±0.77**	7.36
50	88.94±1.16	**94.18±0.48**	5.89
70	90.37±0.93	**95.07±0.86**	4.84
80	91.25±1.56	**95.4±0.87**	4.55
90	91.15±1.73	**96±1.80**	5.32

4.2 Iterative-Higher-Orders Semantic Kernel (IHOSK)

We propose a semantic kernel for Support Vector Machines (SVM) that takes advantage of higher-order relations between the words and between the documents. Conventional approach in text categorization systems is to represent documents as a “Bag of Words” (BOW) in which the relations between the words and their positions are lost. Additionally, traditional machine learning algorithms assume that instances, in our case documents, are independent and identically distributed. This approach simplifies the underlying models, but nevertheless it ignores the semantic connections between words as well as the semantic relations between documents that stem from the words. In this study [92], we improve the semantic knowledge capture capability of a previous work in [18], which is called χ-Sim algorithm and use this method in the SVM as a semantic kernel. The proposed approach is evaluated on different benchmark textual datasets. Experiment results show that classification performance improves over the linear kernel which is one of the state-of-the-art algorithms for text classification [92].

4.2.1 Methodology

In our approach, D_t is the data matrix having *r* rows (documents) and *c* columns (words) formed from the training set. In this matrix d_{ij} shows the occurrence frequency of the j^{th} word in the i^{th} document; $d_i = [d_{i1} .. d_{ic}]$ is the row vector representing the document *i* and $d_j = [d_{1j} .. d_{rj}]$ is the column vector corresponding to word *j*.

We also tried several term weighting methods. First of them is TF-IDF which is a statistical measure used to evaluate the importance of a word for a document in a corpus as mentioned in Chapter 2.Another term weighting approach we investigated is from Dumais’s research in [91]. In this approach, terms are represented in a document after multiplying by a value that is the global weight of the term in the whole corpus. The local weight of a term *t* in a document *d* is calculated as taking the log value of the total frequency of *t* in *d*. The global weight of a term is the entropy of that term in the corpus and according to [91] the entropy equals:

$$Entropy(t,d) = 1 - \sum_{i=1}^{N} \frac{p_{td}\log(p_{td})}{\log(N)} \tag{4.9}$$

where N is the number of documents and p_{td} equals the number of times that t occurs in d divided by the total number of times that t occurs.

However, since we get better accuracies for linear kernel with the only TF schema without any weighting, we use TF instead of TF-IDF or Entropy weighting approaches in our experiments for both linear kernel and our algorithm.

We use the documents in our training corpus for χ-Sim's SC and SR similarity matrix calculations. We calculate up to four iterations by using the following equations:

$$SR_t = D\ SC_{t-1}\ D^T \otimes NR \text{ with } NR_{i,j} = \frac{1}{|d_i||d_j|} \qquad (4.10)$$

$$SC_t = D^T\ SR_{t-1}\ D \otimes NC \text{ with } NC_{i,j} = \frac{1}{|d_i||d_j|} \qquad (4.11)$$

where D is the document by term matrix, D^T is the transpose of D matrix, SR is the row (document) similarity matrix, SC is the column (word) similarity matrix, NR and NC are row and column normalization matrices and $\otimes$ denotes Hadamard multiplication, respectively. In a Hadamard product A=B$\otimes$C, the elements $a_{i,k}$ of matrix A are defined as: $a_{i,k}= b_{i,k}.\ c_{i,k}$.

Similar to [18], we calculate SR_0, SC_0, SR_1, SC_1, SR_2, SC_2, SR_3, SC_3, SR_4, SC_4 matrices and after that we use these SC matrices, which contain iterative higher-order relations between terms, into our kernel by using Eq. (4.12):

$$k_{IHOSK}(d_1, d_2) = d_1 S S^T d_2^T \qquad (4.12)$$

where $k_{IHOSK}(d_1, d_2)$ is the similarity value between documents d_1 and d_2,S is a semantic matrix which is gathered from the previously mentioned calculations of $SC_2$and d_1 and d_2 are term-frequency vectors of the documents. The S is a semantic matrix is based on iterative higher-order paths between documents and between terms. This kernel function means that the transformation of a document vector from input space to a feature space can be done by multiplying it with a semantic matrix as given in Eq. (4.13):

$$\phi(d_1) = d_1 S \text{ and } \phi(d_2) = S^T d_2^T \qquad (4.13)$$

where $\phi(d_1)$ and $\phi(d_2)$ are the transformations of document vectors $d_1$and d_2 from their original input space into the feature space as required in the definition of kernel which is mentioned in Section 2.

After performing experiments up to four iterations of *SC* matrices, we conclude that the best results are obtained with the second iteration matrices (SR_2, SC_2). The following experimental results reflect the results of our approach using these matrices.

Since we work with textual datasets which are high dimensional and highly sparse, we think that it is possible to benefit from normalization methods which could be applied on the similarity matrices. We also experimented with different normalization methods as mentioned in Section 4.1. We obtained best accuracy results with length normalization which is defined in Eq. (4.14).

$$\forall i, j \in 1..r \quad N\text{-IHOSK}(d_i, d_j) = \frac{\text{IHOSK}(d_i, d_j)}{|d_i|.|d_j|} \qquad (4.14)$$

where r is the number of documents in our corpus, IHOSK is similarity value between documents d_i and d_j, N-IHOSK is the normalized similarity value of these documents d_i and d_j and $|d_i|$ and $|d_j|$ are the lengths of these documents depending on the number of terms they have, respectively. Then, we use this kernel function in SVM by plugging in the SMO WEKA's implementation[85]. In other words we built such a kernel function that is directly applicable in Platt's SMO [86] learner.

4.2.2 Experimental Results and Discussion

According to Table 4.6, N-IHOSK outperforms our baseline kernel (linear kernel) by extensive boundaries in all training set percentages. For instance at training levels 30%, 50% and 70% the accuracies of N-IHOSK are 94.31%, 94.97% and 95.35% while the accuracies of linear kernel are 86.73%, 88.94% and 90.37% ,respectively. The performance gain is obvious at all training set levels. It is important to note that high performance gains are especially visible at low training set levels. For instance at training levels 5%, and 10% N-IHOSK outperforms linear kernel with the gains of 18.64% and 16.25%, respectively. As mentioned above, this performance is of great importance since usually it is difficult and expensive to obtain labeled data in real world applications.

Table 4.6Accuracy of N-IHOSK and linear kernel on 20 Newsgroups-Sciencedataset with varying training set size

TS %	linear	N-IHOSK	Gain
5	70.93±3.89	**84.15±2.87**	18.64
10	77.74±3.52	**90.37±0.81**	16.25
30	86.73±1.32	**94.31±1.09**	8.74
50	88.94±1.16	**94.97±0.90**	6.78
70	90.37±0.93	**95.35±0.88**	5.51
80	91.25±1.56	**96.23±1.19**	5.46
90	91.15±1.73	**96.85±1.70**	6.25

Table 4.7Accuracy of N-IHOSK and linear kernel on20 Newsgroups-Politics dataset with varying training set size

TS %	linear	N-IHOSK	Gain
5	78.33±3.40	**82.27±4.60**	5.03
10	84.66±2.09	**88.61±2.1**	4.67
30	91.98±1.24	**93.61±1.08**	1.77
50	91.21±0.89	**93.55±3.58**	2.57
70	92.29±1.22	**93.24±3.08**	1.03
80	93.7±0.79	**95.3±1.82**	1.71
90	93.69±2.04	**95.8±2.28**	2.25

Table 4.8Accuracy of N-IHOSK and linear kernel onWEBKB5dataset with varying training set size

TS %	linear	N-IHOSK	Gain
5	72.77±1.43	**76.12±1.39**	4.60
10	79.12±2.18	**82.41±2.32**	4.16
30	86.10±1.52	**88.27±1.62**	2.52
50	90.16±1.11	**91.89±1.08**	1.92
70	90.60±1.93	**92.31±1.41**	1.89
80	91.00±1.45	**93.10±1.77**	2.31
90	91.93±2.52	**93.13±1.54**	1.31

On 20 Newsgroups-Politics dataset, N-IHOSKproduces better classification accuracies than linear kernel in all of the training levels which can be observable from Table 4.7.

Same trend can be seen for WEBKB5 dataset which has a highly skewed class distribution. In this dataset our algorithm N-IHOSK outperforms linear kernel. This can be seen in Table 4.8.

For us one of the best results is observed on Mini-newsgroups dataset.This dataset has the largest number of classes. Again in all training levels starting from 5% up until 90% N-IHOSK gives higher accuracies than other kernels. This can be seen from Table 4.9. This is especially obvious at 5% training level; the performance gain of N-IHOSKon linear kernel is 17.79%

Table 4.9Accuracy of N-IHOSK and linear kernel on Mini-newsgroupsdataset with varying training set size

TS %	**linear**	**N-IHOSK**	**Gain**
5	52.03±5.95	**61.29±1.03**	17.79
10	59.31±4.58	**64.15±0.54**	8.16
30	72.61±4.23	**75.51±0.31**	4.00
50	76.02±4.24	**79.24±0.31**	4.24
70	77.61±2.76	**79.73±0.45**	2.73
80	80.70±2.20	**83.05±0.58**	2.91
90	83.25±4.05	**85.38±1.28**	2.56

The particularly high accuracies of the proposed method on 20 Newsgroups-Science dataset may be explained with the less average sparsity of the documents of this dataset compare to the other datasets. It is possible that having more terms in documents of this dataset give us the opportunity to generate more higher-order paths between documents.

At small training data levels first-order methods give zero as the similarity of two documents that do not contain common words. But by the use of higher-order paths the similarity between those two instances can be larger than zero. We think that this is the main reason that the difference between N-IHOSK and linear kernel which is most visible at small training levels like 5% and 10%. Through the experiments we observed remarkable gains such as 18.64%, 16.25%, and 17.79% at only using 5% and 10% of the labeled data as training set. This has important implications on real world applications where the labeled data is generally difficult to obtain. In many real world applications serious costs are associated with the labeling of the data.

4.3 Higher-Order Term Kernel (HOTK)

In this study [93], we present a simple semantic kernel for SVM algorithm. This kernel uses higher-order relations between terms in order to incorporate semantic information into the SVM. This is an easy to implement algorithm which forms a basis for future improvements. We perform a serious of experiments on different well known textual datasets. Experiment results show that classification performance improves over the traditional kernels used in SVM such as linear kernel which is commonly used in text classification[93].

4.3.1 Methodology

In our proposed method, D_{train} is the data matrix having r rows (documents) and t columns (words) formed using the training set. In this matrix d_{ij} represents the occurrence frequency of the j^{th} word in the i^{th} document; di = [$d_{i1},\ldots,d_{it}$] is the row vector showing the document *i*and dj = [$d_{1j},\ldots,d_{rj}$] is the column vector belongs to word *j*.

It is important to note that binary term occurrences are used in the premier studies which use higher-order paths between terms since it simplifies the definition and counting of the higher-order paths. However, in this study [93], we experiment with term frequencies (TF). This is similar to the initial attempt to use term frequencies in[17].

We use the training set to extract higher-order paths between terms. The *S* matrix which shows the amount or weight of higher-order (second-order in this case) relations between terms is obtained by using the formula in Eq. (4.15). This approach is motivated by algorithm which is explained in[17]. However, in this study [93], we are using term frequencies instead of binary term occurrences and we are not filtering any paths.

$$S = D_{train}^{T} D_{train} \tag{4.15}$$

whereD_{train} is document by term matrix of the training set *S* is a symmetric square matrix whose dimensions are the number of the terms in the training set. The *S* matrix displays the first-order relations, in other words just co-occurrences of the terms. In order to get higher co-occurrence relations or in other words higher-order paths we multiply the *S* by itself. For instance, the square of *S* reveals the second-order relations between the terms.

Since the second-orderpaths reveal latent semantic relations [15] we use the following Eq. (4.16) as a simple semantic kernel.

$$k_{HOT-K}(d_1, d_2) = d_1 S \ S^T d_2^T \qquad (4.16)$$

The proposed kernel function in Eq. (4.16) means that the transformation of a document vector from input space to a semantic feature space can be accomplished by multiplying it with a semantic matrix as shown in Eq. (4.17).

$$\phi(d_1) = d_1 S \text{ and } \phi(d_2) = S^T d_2^T \qquad (4.17)$$

where $\phi(d_1)$ and $\phi(d_2)$ vectors are the transformations of documents d_1 and d_2 vectors from their original input space into the feature space as required in the definition of kernel which is mentioned in Section 2.

We also experimented with different normalization methods as mentioned in Section 4.1. We obtained best accuracy results with length normalization which is defined in Eq. (4.18):

$$\forall i, j \in 1..r \quad k_{norm}(d_i, d_j) = \frac{k(d_i, d_j)}{|d_i| . |d_j|} \qquad (4.18)$$

where $|d_i|$ and $|d_j|$ are the lengths of these documents measured by the sum of the term occurrences.

4.3.2 Experimental Results and Discussion

According to our experiments HOTKdemonstrates a notable performance on 20 Newsgroups-Science dataset, which can be seen in Table 4.10. HOTKoutperforms our baseline kernel (linear kernel) in all training set percentages. The performance gain is specifically obvious at low training set levels. For instance, at training levels 5% and 10% HOTKoutperforms linear kernel with the gains of 7.26% and 5.77% on linear kernel respectively. 20 Newsgroups-Science dataset is also used in our previous studies[92], [97]. Therefore we use this dataset to compare the results of HOSK [97] and IHOSK [92] with HOTK[93]. Although the HOTK be able to outperform the baseline (linear kernel), the performance of IHOSK is superior to the HOSK and HOTK. However, the complexity of IHOSK is much higher than the previous works such as HOSK, and the proposed work of the HOTK. This prevents the IHOSK to be applied on large datasets. HOSK also performs slightly better than HOTK but it is based on the higher-order paths between documents. The semantic relations between the documents are not as clear as the relations between the terms. HOTK is our first attempt to use the higher-order paths between terms as a semantic kernel for SVM. Using higher-order paths between terms instead of between documents (as in HOSK) or both the documents and terms (as in IHOSK) forms a foundation that is open to several improvements. For instance HOTK can easily be combined with other term based semantic kernels such as the ones using WordNet or Wikipedia. Furthermore, it will be much easier to apply different path filters and normalizations based on the role of terms in different classes and observe their affects.

Table 4.10 Accuracy of HOTK and other kernels on 20 Newsgroups-Science dataset with varying training set size

TS %	**Linear**	**HOSK**	**IHOSK**	**HOTK**	**Gain**
5	71.44±4.30	85.69±1.80	**90.37±0.81**	76.63±2.67	7.26
10	77.97±3.73	87.87±1.34	**94.31±1.09**	82.47±2.02	5.77
30	86.73±1.32	93.11±0.77	**94.97±0.90**	89.24±0.74	2.89
50	88.94±1.16	94.18±0.48	**95.35±0.88**	90.84±1.12	2.14
70	90.58±0.93	95.07±0.86	**96.23±1.19**	92.06±1.28	1.63
80	91.33±1.41	95.40±0.87	**96.85±1.70**	93.38±1.43	2.24
90	91.40±1.56	**96.00±1.80**	94.31±1.09	94.2±1.36	3.06

For the remaining datasets we report the results of HOTK compared to the baseline kernel. 20 Newsgroups-Politics is an exceptional dataset in terms of the performance of HOTK. We only see improvements at very low training set percentages. This may due to the size of the dataset. 20 Newsgroups-Politics is our smallest dataset with 3 classes and 1500 documents. We observe that the discussions are centered on a smaller number of topics compare to the other datasets. In our opinion in this dataset, the classes are easier to discriminate, giving more advantage to the document based methods.

For 20 Newsgroups-Comp dataset, HOTK outperforms linear kernel in all training levels. This can be seen from Table 4.12. 20 Newsgroups-Comp is a larger dataset than the 20 Newsgroups-Politics. It has five classes. As expected, the performance improvement is most visible in small training set levels which can be seen from Table

4.12. For Mini-newsgroups dataset, HOTK outperforms linear kernel in almost all of the training levels. This can be seen from Table 4.13.

Table 4.11 Accuracy of HOTK and linear kernel on 20NewsPolitics dataset with varying training set size

TS %	**Linear**	**HOTK**	**Gain**
5	79.01±2.65	**80.72±1.56**	2.16
10	84.69±1.24	**84.89±2.15**	0.24
30	**92.04±1.06**	88.31±1.22	-4.05
50	**93.73±0.57**	90.29±0.79	-3.67
70	**94.55±1.21**	90.15±1.15	-4.65
80	**94.03±0.91**	92.50±1.60	-1.63
90	**94.86±1.26**	92.46±2.01	-2.53

Table 4.12Accuracy of HOTK and linear kernel on 20 Newsgroups-Comp dataset with varying training set size

TS %	**Linear**	**HOTK**	**Gain**
5	56.75±4.72	**60.22±3.00**	6.11
10	65.45±2.77	**66.70±1.14**	1.91
30	75.38±2.12	**75.97±1.04**	0.78
50	77.89±1.60	**78.68±0.71**	1.01
70	79.63±1.59	**79.97±1.18**	0.43
80	79.00±2.25	**80.38±1.85**	1.75
90	81.40±2.47	**81.52±1.46**	0.15

Table 4.13Accuracy of HOTK and linear kernel on Mini-newsgroups dataset with varying training set size

TS %	**Linear**	**HOTK**	**Gain**
5	**56.75±4.72**	49.69±5.64	-12,44
10	65.45±2.77	**66.24±3.81**	1,21
30	75.38±2.12	**81.82±2.04**	8,54
50	77.89±1.60	**85.54±1.20**	9,82
70	79.63±1.59	**87.28±1.13**	9,61
80	79.00±2.25	**88.15±1.58**	11,58
90	84.65±2.48	**88.10±2.80**	4,08

CHAPTER 5

CORPUS-BASED SEMANTIC KERNELS BY USING CLASS-BASED MEANING AND WEIGHT VALUES OF TERMS

5.1 Class Meanings Kernel (CMK)

In this study[94], we propose a novel approach for building a semantic kernel for SVM, which we name Class Meaning Kernel (CMK). The suggested approach smoothes the terms of a document in BOW representation (document vector represented by term frequencies) by class-based meaning values of terms. This in turn, increases the importance of significant or in other words meaningful terms for each class while reducing the importance of general terms which are not useful for discriminating the classes. This approach reduces the disadvantages of BOW and improves the prediction abilities in comparison with standard linear kernels by increasing the importance of class specific concepts which can be synonymous or closely related in the context of a class. The main novelty of this approach is the use of this class specific information in the smoothing process of the semantic kernel. The meaning values of terms are calculated according to the Helmholtz principle from Gestalt theory [99], [100], [101], [102] in the context of classes as mentioned in Section 2.4.

5.1.1 Methodology

In our study, we use the general form of kernel function which is given in Eq. (2.8). The simplest form of kernel function, namely linear kernel is formulated in Eq. (2.5). But as it is criticized in Section 2 linear kernel is a simple dot product between the features of text documents. It produces a similarity value of two documents only proportional to the number of shared terms. Combined with the highly sparse representation of the textual data, this may yield a significant problem especially when two documents are written about the same topic using two different sets of terms which are actually semantically very close as it is mentioned in the Section 2.2. Also, in cases where training data is scarce there will be serious problems to detect reliable patterns between documents. This means that using only simple dot product to measure similarity between documents will not always give sufficiently accurate similarity values between documents. Additionally, for a better classification performance it is inevitably required to discount general words and emphasize more importance on core words (which are closely related to the subject of that class) as it is analyzed in [3]. In order to overcome these mentioned drawbacks, semantic smoothing kernels encode semantic dependencies between terms [11], [41], [103],[104]. We also incorporated additional information of terms other than their simple frequencies as in our previous studies [92], [93], [97]in which we take advantage of higher-order paths between words and/or documents. In those studies we showed that the performance difference between first-order and higher-order representation of features. In this approach we investigate the use of a new type of semantic smoothing kernel for text.

Figure 5.1 demonstrates the architecture of the suggested semantic kernel. This system mainly consists of four independent modules: preprocessing, meaning calculation, building semantic kernel, and classification. Preprocessing is the step that involves the conversion of input documents into formatted information. This step includes stemming and stopword filtering. In meaning calculation step, the meaning values of the terms according to the classes are calculated based on Eq. (2.32). Then we construct our proposed kernel, namely CMK, in the step for building semantic kernel. Finally, in the classification step SVM classifier builds a model in the training phase and this model is then applied to the test examples in the test phase.

Clearly, the main feature of this system is that it takes advantages of the meaning calculation in kernel building process, in order to reveal semantic similarities between terms and documents by smoothing the similarity and the representation of the text documents. Meaning calculation is based on Helmholtz principle from Gestalt theory. As mentioned in Section 2.4, this meaning calculations have been applied to many domains in previous works (for example information extraction [105], text summarization [101], rapid change detection in data streams [99], and keyword

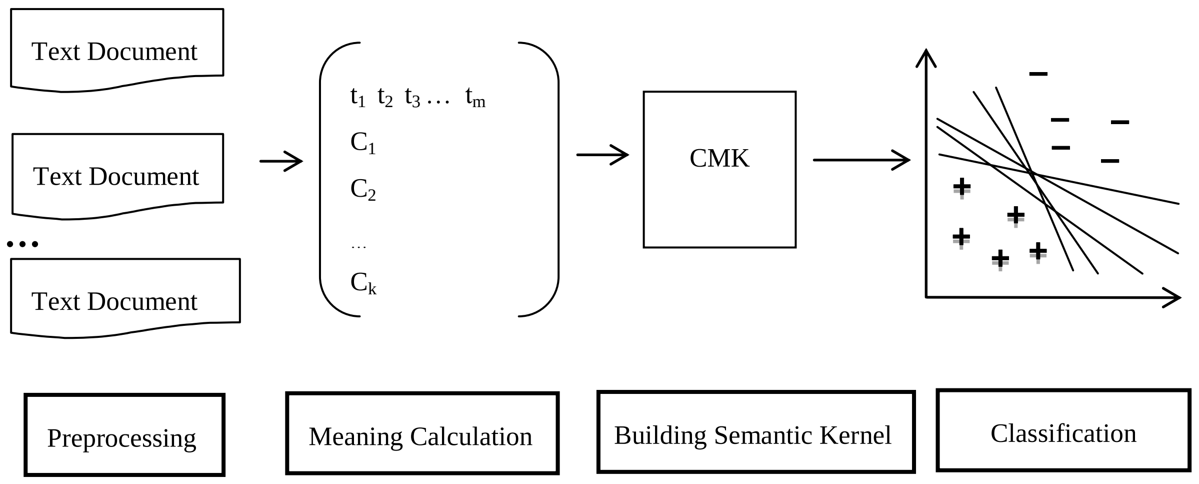


Figure 5.1The architecture of CMK System

extraction). In these studies a text document is modelled by a set of meaningful words together with their meaning scores. A word is considered meaningful or important if the term frequency of a word in a document is unexpected if we consider the term frequencies of this word in all the documents in our corpus. The method can be applied on a single document or on a collection of documents to find meaningful words inside each part or context (paragraphs, pages, sections or sentences) of a document or a document inside of a collection of documents[102]. Although meaning calculation has been used in several domains, to the best of our knowledge, our work is the first to apply this technique to kernel function.

In our methodology D_{train} is the data matrix of training set having r rows (documents) and t columns (terms). In this matrix d_{ij}stands for the occurrence frequency of the j^{th} word in the i^{th} document; $d_i = [d_{i1},\ldots,d_{it}]$ is the document vector showing the document i and $d_j = [d_{1j},\ldots,d_{rj}]$ is the term vector belonging to word j, respectively. To enrich D_{train}, with semantic information, we build the class-based term meaning matrix M using meaning calculations given in Eq. (2.32). The M matrix shows the meaningfulness of the terms in each class. Based on M we calculate S matrix in order to reveal class based semantic relations between terms. Specifically, the ij^{th} element of S quantifies the semantic relatedness between terms $t_i$and t_j.

$$S = MM^T \tag{5.1}$$

In our system *S* is a semantic smoothing matrix to transform documents from input space to feature space. Thus, *S* is a symmetric term-by-term matrix. Mathematically, the kernel value between two documents is given as

$$k_{CMK}(d_1, d_2) = d_1 \; S \; S^T d_2^T \qquad (5.2)$$

where $k_{CMK}(d_1, d_2)$ is the similarity value between documents d_1 and d_2, *S* is the semantic smoothing matrix. In other words, here *S* is a semantic proximity matrix which derives from the meaning calculations of terms and classes.

If a word occurs only once in a class then its meaning value for that class is zero according to Eq. (2.32). If a word does not occur at all in a class, it gets minus infinity based on Eq. (2.32) as a meaning value for that class. In order to make calculations more practical we assign the next smallest value to that word according to the range of meaning values we get for all the words in our corpus. After all calculations we get *M* as a term-by-class matrix which includes the meaning values of terms in all classes of the corpus. We observe that these meaning values are high for those words that allow us to distinguish between classes. Indeed terms semantically close to the theme discussed in the documents of that class gain the highest meaning values in the range. In other words semantically related terms of that class, i.e. “core” words like it is mentioned in[3], gain importance while semantically isolated terms, i.e. “general” words lose their importance. So terms are ranked based on their importance. For instance, if the word “data” is highly present while the words “information” and “knowledge” are less, the application of semantic smoothing will increase the values of the last two terms because “data”, “information” and “knowledge” are strongly related concepts. The new encoding of the documents is richer than the standard TF-IDF encoding since; additional statistical information that is directly calculated from our training corpus is embedded into the kernel. In other words transformations in Eq. (5.2) smooth the basic term vector representation using semantic ranking while passing from the original input space to a feature space through kernel transformation functions $\phi(d_1)$ and $\phi(d_1)$ for the documents d_1 and $d_2$respectively:

$$\phi(d_1) = d_1 S \text{ and } \phi(d_2) = S^T d_2^T \qquad (5.3)$$

As mentioned in[106], the presence of S in Eq. (5.3) changes the orthogonality of the vector space model, as this mapping introduces term dependence. Documents can be seen as similar even if they do not share any terms by eliminating orthogonality.

Also as it is mentioned in [99], meaning calculation automatically filters stop words by assigning them very small amounts of meaning values. Let us consider the following two cases, which are represented in Table 5.1. According toTable 5.1, it is understood that $t_1$and t_2 occurred in one or more documents of c_1, not in remaining classes; c_2, $c_3$and c_4, respectively. In other words t_1 and $t_2$are critical words of the topic discussed in c_1; getting high meaning values according to Eq. (2.32); since the frequency of a term in a class,m is inversely proportional to the NFA. According to Eq. (2.32), in such a case the number of times that word occurred in the whole corpus (k) is larger when the times of that word's occurrence in a class (m) is smaller NFA calculation directly gives a larger negative value which will yield a larger positive value. In other words, according to the spirit of meaning value calculation, the more a word occurred in only a specific class the higher meaning value it gets, and conversely the more a word occurred in all classes the less meaning value it gets. This statement can also be represented withTable 5.1, since t_1 and t_2 occurred in only c_1 while $t_3$and$t_4$occurred in every classes of the corpus. It is highly possible that these two words, t_3 andt_4, are in the type of "general" words since they are seen in every class of the corpus.

Table 5.1Term frequencies in different classes

	c_1	**c_2**	**c_3**	**c_4**
t_1	1	0	0	0
t_2	1	0	0	0
t_3	1	1	1	1
t_4	1	1	1	1

5.1.2 Experimental Results and Discussion

CMK outperforms our baseline kernel clearly in almost all training set percentages on 1150Haber dataset. This can be observed from Table 5.2. CMK demonstrates much better performance than linear kernel on this dataset, in all training set percentages except 5% and 10%. The performance gain is specifically obvious starting from 30% training set percentage. For instance at training set percentages 30%, 50% , 70% , 80% and 90% the accuracies of CMK are 91.49%, 93.81%, 93.94%, 93.09% and 93.74% while the accuracies of linear kernel are 88.55%, 89.72%, 91.59, 92.30% and 91.83%;

respectively. CMK also has better performance than our previous semantic kernels IHOSK, and HOTK at training set percentages between 30% and 90% as shown in Table 5.2. The highest gain of CMK over linear kernel on this dataset is at 50% training set percentage which is 4.56%. Additionally, according to Table 5.2 we can conclude that the performance differences of CMK while passing from one training set percentage to another are compatible with the term coverage ratios at those training set percentages. For instance at training set percentage 30%, term coverage jumps to 71.56% from its previous value at 10% that is 54.60%. Similar behavior can be observed at performance of CMK while going through 10% training set percentage to 30% training set percentage; where it generates the accuracies 72.07% and 91.49%; respectively. This means an accuracy change of 19.42% between 10% and 30% training set percentages. Furthermore we also performed our experiments for linear kernel and CMK on 1150Haber dataset without IG. According to Table 5.2 both linear kernel and CMK generate better classification accuracies on 1150Haber dataset when IG (selection of 2000 attributes with IG) is used in compare to the case of without IG. This experimental result shows us the necessity of doing preprocessing (e.g. attribute selection and filtering stop words and rare words).

Table 5.2 Accuracy of CMK and other kernels on 1150Haber dataset with varying training set size

TS %	**Linear**	**Linear (without IG)**	**IHOSK**	**HOTK**	**CMK**	**CMK (without IG)**	**Gain**
5	72.82±4.68	66.51±4.58	**74.54±1.75**	73.34±1.71	55.15±8.63	37.76±7.49	-24.27
10	80.51±2.68	76.93±1.68	**80.74±1.17**	79.97±1.24	72.07±4.48	42±4.99	-10.48
30	88.55±1.34	85.84±1.56	90.12±1.69	89.47±1.76	**91.49±2.12**	80.82±9.45	3.32
50	89.72±1.13	88.54±1.23	92.73±1.11	90.83±0.12	**93.81±1.31**	87.84±3.27	4.56
70	91.59±1.06	89.51±1.79	93.14±0.87	91.76±1.86	**93.94±1.08**	90.9±1.23	2.57
80	92.30±2.89	90.13±1.94	93.75±1.98	92.15±0.36	**93.89±2.91**	91.38±1.10	0.86
90	91.83±3.18	91.13±2.31	94.25±1.72	92.70±1.65	**94.54±1.82**	92.78±2.81	2.08

Table 5.3 Accuracy of CMK and other kernels on 20 Newsgroups-Science dataset with varying training set size

TS %	Linear	IHOSK	HOTK	CMK	Gain	Term Coverage
5	**71.44±4.3**	84.15±2.87	76.63±2.67	64.51±4.86	-9.70	63.99
10	77.97±3.73	90.37±0.81	82.47±2.02	**82.19±3.58**	5.41*	82.28
30	86.73±1.32	94.31±1.09	89.24±0.74	**95.07±0.87**	9.62*	98.01
50	88.94±1.16	94.97±0.90	90.84±1.12	**96.71±0.61**	8.74*	99.90
70	90.58±0.93	95.35±0.88	92.06±1.28	**97.12±0.59**	7.22*	99.99
80	91.33±1.41	96.23±1.19	93.38±1.43	**97.60±0.66**	6.87*	100.00
90	91.40±1.56	96.85±1.70	94.20±1.36	**97.75±0.89**	6.95*	100.00

Table 5.4Accuracy of CMK and other kernels on IMDB dataset with varying training set size

TS %	Linear	IHOSK	HOTK	CMK	Gain	Term Coverage
5	76.85±1.31	76.98±1.14	74.21±0.24	**77.84±2.99**	1.29	48.00
10	82.99±1.76	82.55±2.32	82.23±0.42	**84.51±1.45**	1.83	61.51
30	85.57±1.65	87.16±1.64	85.63±1.69	**90.54±0.65**	5.81*	86.35
50	88.46±1.89	89.40±1.91	87.20±0.33	**92.30±0.59**	4.34	95.91
70	89.93±1.18	91.31±0.87	90.41±0.55	**93.23±0.70**	3.67	99.17
80	90.65±1.09	92.38±1.43	91.37±0.98	**93.43±0.94**	3.07	99.71
90	91.75±1.14	92.63±1.19	91.59±0.27	**93.65±0.37**	2.07	99.98

CMK outperforms our baseline kernel clearly in almost all training set percentages on 20 Newsgroups-Science dataset. This can be observed from Table 5.3. CMK demonstrates much better performance than linear kernel on this dataset, in all training set percentages except 5%. The performance gain is specifically obvious starting from 10% training set percentage. For instance at training set percentages 30%, 50% , 70% , 80% and 90% the accuracies of CMK are 95.07%, 96.71%, 97.12%, 97.6% and 97.75% while the accuracies of linear kernel are 86.73%, 88.94%, 90.58, 91.33% and 91.4%%; respectively. CMK also has better performance than our previous semantic kernels IHOSK, and HOTK at training set percentages between 30% and 90% as shown in Table 5.3. The highest gain of CMK over linear kernel on this dataset is at 30% training set percentage which is 9.62%. Also it should be noted that, there is a performance gain of

CMK over linear kernel 5.41% at training set percentage 10%, which is of great importance since usually it is difficult and expensive to obtain labeled data in real world applications. Additionally, according to Table 5.3we can conclude that the performance differences of CMK while passing from one training set percentage to another are compatible with the term coverage ratios at those training set percentages. For instance at training set percentage 30%, term coverage jumps to 98.01% from its previous value at 10% that is 82.28%. Similar behavior can be observed at performance of CMK while going through 10% training set percentage to 30% training set percentage; where it generates the accuracies 82.19% and 95.07%; respectively. This means an accuracy change of 12.88% between 10% and 30% training set percentages.

Additional to CMK, that is calculated with Eq. (5.4) and Eq. (5.5) we also built a second-order version of CMK with the name Second-Order Class Meaning Kernel (SO-CMK) with the following equation:

$$k_{SO-CMK}\ (d_1, d_2) = d_1\ S^2\ S^2 d_2^T \qquad (5.4)$$

where *S* is our term-by-term meaning matrix that is also used for CMK. Transformations are done with;

$$\phi(d_1) = d_1\ S^2 \text{ and } \phi(d_2) = S^2\ d_2^T \qquad (5.5)$$

where $\phi(d_1)$ and $\phi(d_1)$ are transformation functions of kernel from input space into feature space for the documents *d₁* and *d₂*, respectively. In other words, here *M* is asemantic proximity matrix of terms and classes which shows semantic relations between terms. In this case semantic relation between two terms is composed of corresponding class based meaning values of these terms for all classes. So if these two terms are important terms in the same class then the resulting semantic relatedness value will be higher. In contrast to the other semantic kernels that makes use of WordNet or Wikipedia in an unsupervised fashion, CMK directly incorporates class information to the semantic kernel. Therefore, it can be considered as a supervised semantic kernel.

We also recorded and compared the total kernel computation time of our previous semantic kernels IHOSK and HOTK and CMK. All the experiments presented here are carried on our experiment framework, Turkuaz, which directly uses WEKA[85] on a computer with two Intel(R) Xeon(R) CPUs at 2.66 GHz with 64 GB of memory. Our semantic kernel's computation time on each dataset is recorded in terms of seconds and they are proportionally converted into percentages by making the longest run time 100.According to this conversion, for instance on 20 Newsgroups-Science dataset; IHOSK[92], SO-CMK, CMK and HOTK [93] estimates the following time units in order; 100, 55, 32, and 27, respectively, which is shown inFigure 5.2.

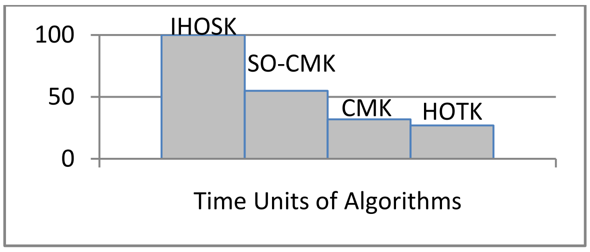


Figure 5.2The total kernel computation time units of IHOSK, SO-CMK, CMK and HOTK on 20 Newsgroups-Science dataset at 30% training set size

These values are not surprising since the complexity and running time analysis supports them. In IHOSK [92]there is an iterative similarity calculation between documents and terms, which completes totally in four steps including corresponding matrix calculations as in shown in Eq. (4.10) and Eq. (4.11). As it is discussed in [18] producing the similarity matrix (SC_t) has overall complexity O(tn^3) where t is the number of iterations and n is the number of training instances. Since in our experiments we fixed t=2 we obtain O($2n^3$) complexity. On the other hand HOTK[93]has complexity O(n^3) as it can be noted from Eq. (4.16). CMK also has a complexity of O(n^3) like HOTK, but additional to the calculations made for HOTK, CMK has a phase of calculating meaning values which makes CMK run slightly longer than HOTK as shown in Figure 5.2. Moreover, SO-CMK includes additional matrix multiplications as a result it runs longer than CMK. Since the IHOSK involves much more matrix multiplications than both HOTK and the proposed work of the CMK, it runs almost three times longer than the proposed approach on a relatively small dataset with 2,000 documents and 2,000 attributes.

We also compare CMK with a kernel based on a similar method of TF-ICF which is explained in Section 2.3. We compare the results of TF-ICF to CMK with Eq. (2.16) which indeed a supervised approach as mentioned in Section 2.4. Additionally we also created an unsupervised version of Meaning kernel, Unsupervised Meaning Kernel (UMK), by using a single document as our context (the P value in Eq. (2.32)) instead of using a class of documents. This introduces an unsupervised behavior into CMK since our basic unit is not class but instead a single document. The results are shown inFigure 5.3. The CMK has much better performance than both UMK and TF-ICF in almost all training set percentages except 10%. Starting from training set percentage 10% the difference between the performance of CMK and the other two algorithms start to increase.

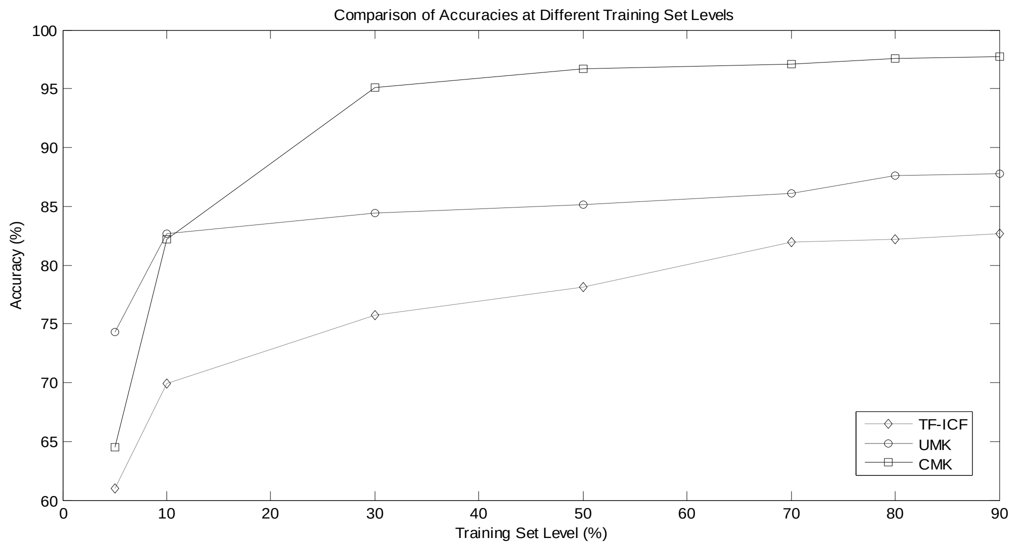


Figure 5.3The Comparison of the accuracies of TF-ICF, UMK and CMK at different training set percentages on 20 Newsgroups-Science dataset

According to our experiments, the CMKdemonstrates a notable performance gain on the IMDB dataset, which can be seen inTable 5.4. The CMKoutperforms our baseline, linear kernel, in all training set percentages also making a significant difference at training set percentage 30% based on Students t-Tests results. In training set percentage 30% the performance of the CMK is 90.54% while the performance of linear kernel is only 85.57%. It is also very promising to see that the CMK is superior to both linear kernel and our previous algorithms IHOSK [92] and HOTK [93] throughout all training set percentages.

Table 5.5presents the experiment results on the 20 Newsgroups-Politics dataset. In this dataset, the CMK's performance is higher than linear kernel's in all training set percentages except 5% and 10%. Furthermore, the CMK is performs better than both IHOSK and HOTK in almost all training set percentages except 5% and 10%. Only in training set percentages 5% and 10%, the IHOSK gives better accuracy than the CMK.

For 20 Newsgroups-Compdataset, the CMKoutperforms linear kernel in all training set percentages except 5% as shown in Table 5.6. The CMK yields higher accuracies compared to linear kernel, IHOSK and HOTK. The differences between CMK and linear kernel are statistically significant according to Student's t-test at training levels 10%, 30%, 50%, 70%, 80%, and 90%.

Experiment results on 20 Newsgroups-Religiondataset are presented in Table 5.7. These results show that the CMK has superiority starting from 30% training set percentage among all of the other kernels. For instance at training set percentage 30% CMK's gain over linear kernel is 8.58%. Also, in training set percentages 30% and 50%, the CMK shows a significant improvement over linear kernel.

Table 5.8presents the experiment results on Mini-newsgroups dataset. According to these results the CMK outputs better accuracy than linear kernel at training set percentages 30%, 50%, 70%, 80% and 90%. But in overall the CMK is not as good as HOTK on this dataset, which can be explained by the capability of HOTK for capturing latent semantics between documents by using higher-order term co-occurrences as explained in Section 2.2. These latent relations may play an important role since the number of classes is relatively high and the number of documents per class is much smaller yielding a higher sparsity that can be observed from the term coverage statistics.

Since some of the datasets used in this study [94] are also used in [14], we have the opportunity to compare our results with HOSVM. For instance at training level 30%, on 20 Newsgroups-Compdataset; 75.38%, 78.71%, 75.97%, and 84.31% accuracies are gathered by linear kernel, IHSOK, HOTK and CMK as mentioned in above tables and paragraphs. On the same training level HOSVM achieves 78% accuracy according to the Fig. 2(d) in [14]. This comparison shows CMK outperforms HOSVM by approximately 8.28% gain. Actually CMK's superiority on HOSVM is still valid on other datasets such as 20 Newsgroups-Religion, 20 Newsgroups-Science and 20 Newsgroups-Politics, too. For instance on 20 Newsgroups-Politics dataset while HOSVM' performance is about 91%, CMK reaches 96.53% accuracy, which produces a gain of 8.95%. Very similar comparison results can be seen at a higher training level such as 50%.For example the experiment results of 88.94, 92, 94.97, 90.84, 96.71 are achieved by linear kernel, HOSVM, IHSOK, HOTK and CMK at 20 Newsgroups-Science dataset at training level 50%; respectively.

Table 5.5Accuracy of CMK and other kernels on 20 Newsgroups-Politics dataset with varying training set size

TS%	Linear	IHOSK	HOTK	CMK	Gain	Term Coverage
5	79.01±2.65	82.27±4.60	**80.72±1.56**	65.80±3.99	-16.72	58.60
10	84.69±1.24	88.61±2.10	**84.89±2.15**	78.50±6.05	-7.31	75.02
30	92.04±1.06	93.61±1.08	88.31±1.22	**95.03±0.70**	3.25	96.37
50	93.73±0.57	93.55±3.58	90.29±0.79	**96.43±0.58**	2.88	99.43
70	94.55±1.21	93.24±3.08	90.15±1.15	**95.82±0.62**	1.34	99.97
80	94.03±0.91	95.30±1.82	92.50±1.60	**96.73±0.87**	2.87	100.00
90	94.86±1.26	95.80±2.28	92.46±2.01	**96.53±1.57**	1.76	100.00

Table 5.6Accuracy of CMK and other kernels on 20 Newsgroups-Compdataset with varying training set size

TS%	Linear	IHOSK	HOTK	CMK	Gain	Term Coverage
5	56.75±4.72	**68.12±1.04**	60.22±3.00	55.97±5.01	-1.37	48.26
10	65.45±2.77	**72.71±0.43**	66.70±1.14	70.21±3.88	7.27*	65.19
30	75.38±2.12	78.71±0.04	75.97±1.04	**84.31±0.91**	11.85*	91.51
50	77.89±1.60	82.18±1.13	78.68±0.71	**85.02±0.72**	9.15*	98.92
70	79.63±1.59	84.67±2.83	80.97±1.18	**85.60±1.16**	7.50*	99.83
80	79.00±2.25	85.81±0.54	81.58±1.85	**85.78±1.42**	8.58*	99.98
90	81.40±2.47	85.96±0.69	81.32±1.46	**86.00±2.32**	5.65*	100.00

Table 5.7Accuracy of CMK and other kernels on 20 Newsgroups-Religiondataset with varying training set size

TS%	Linear	IHOSK	HOTK	CMK	Gain	Term Coverage
5	74.73±2.47	**77.73±2.47**	65.33±1.70	58.98±7.21	-21.08	41.80
10	80.98±2.69	**81.19±1.92**	72.10±1.95	71.39±7.57	-11.84	59.03
30	83.87±0.78	84.85±1.84	83.50±1.58	**91.07±1.39**	8.58*	88.18
50	88.39±0.93	88.96±2.30	86.19±1.35	**93.04±0.64**	5.26*	96.16
70	89.68±1.41	90.62±1.18	87.26±0.31	**93.47±1.23**	4.23	99.37
80	90.70±1.12	91.00±0.20	88.90±0.24	**93.37±1.68**	2.94	99.8
90	91.65±1.63	91.70±1.73	89.00±2.37	**93.80±2.18**	2.35	99.99

Table 5.8Accuracy of CMK and other kernels on Mini-newsgroups dataset with varying training set size

TS%	Linear	IHOSK	HOTK	CMK	Gain	Term Coverage
5	52.38±5.53	**61.29±1.03**	49.69±5.64	48.89±2.62	-6.66	34.90
10	59.85±3.88	64.15±0.54	**66.24±3.81**	59.53±2.49	-0.53	50.08
30	72.84±3.56	75.51±0.31	**81.82±2.04**	74.24±1.71	1.92	76.16
50	78.87±2.94	79.24±0.31	**85.54±1.20**	79.65±1.64	0.99	87.65
70	80.05±1.96	79.73±0.45	**87.28±1.13**	80.23±1.58	0.22	94.27
80	82.63±1.36	83.05±0.58	**88.15±1.58**	83.53±1.72	1.09	96.22
90	84.65±2.48	85.38±1.28	**88.10±2.80**	85.64±2.87	1.17	98.55

5.2 Class Weighting Kernel (CWK)

In this study[95], we propose a novel approach for building a semantic smoothing kernel which makes use of the class-based term weights to improve the performance of SVM especially for text classification. The proposed approach is called Class Weighting Kernel (CWK). This class-based weighting basically groups terms based on their importance for each class. Consequently it smooths the representation of documents which changes the orthogonality of the vector space model by introducing class-based dependencies between terms. As a result, on the extreme case, two documents can be seen as similar even if they do not share any terms but their terms are similarly weighted for a particular class [95].

5.2.1 Methodology

In our previous studies [92], [93], [94],[97]we take advantage of higher-order paths and meaning calculations. In those studies we show that the performance improvements between first-order and higher-order representation of features [92], [93], [97]and the power of meaning calculations[94]. In this approach we investigate the use of a new type of semantic smoothing kernel for text classification. The main idea behind Class Weighting Kernel (CWK) is to take advantage of the class-based term weighting of terms in the semantic kernel building process by establishing the semantic relations between terms based on their relative weights for classes. This basically gives more importance to core words of each class during the transformation phase of SVM from input space to feature space. Term weighting calculation used in this study[95] is taken from [81] and is motivated by TF-RF [79] and TF-ICF [82][83] as mentioned in Section 2.3. This term-weighting calculation has been applied to feature extraction in previous works [80][81]. In these studies, a text document is represented by terms and their class-based weights. A term has a more discriminative power on a class if it has higher weight for that class. In other words, the more a word occurred in only a specific class the higher its weight gets and conversely the more a word occurred in all the classes it weight gets lower. Although, this class-based weighting calculation has been used in feature extraction domain, to the best of our knowledge, our work is the first to apply this technique to a kernel function.

The VSM represents a document collection by a term-by-document matrix. In the initial step of our methodology a document *d* is represented in the VSM with the following BOW approach:

$$\phi(d) = [tf(t_1,d), tf(t_2,d), tf(t_3,d), tf(t_4,d), tf(t_5,d), \ldots, tf(t_D,d)] \quad (5.6)$$

where tf(t_i,d) is the frequency of term t_i in document *d*, and *D* is the size of the dictionary of the corpus. In above expression $\phi(d)$ represents the document *d* as a TF vector, respectively. This function however, can be any other mapping from a document to its VSM representation (e.g., TF-IDF).

To enrich the BOW representation with semantic information, we build the semantic relatedness matrix S using the class-based term weighting approach. Specifically, the *i, j* element of *S* quantifies the semantic relatedness between terms $t_i$and t_j. The class-based weighting calculations and formulas have been described in detail in the previous section. We take benefits of calculated weights of terms in the mapping schema of our kernel function as

$$S = W\ W^T \quad (5.7)$$

where *W* is a class-based term weighting matrix that is mentioned in Section 2.3 and calculated with Eq. (2.18). W is a term-by-class matrix. In our system S is a semantic smoothing matrix to transform documents from input space to feature space. Thus, S is a symmetric term-by-term matrix. Mapping of document d to new feature space is done in Eq. (5.8).

$$\bar{\phi}(d) = \phi(d)\ S \quad (5.8)$$

Although the feature space defined above can be directly used in many classification methods; in a text classification case where we have high dimensionality with sparsity, it will be helpful to define the feature space implicitly via the kernel function. As it is mentioned in Section 2.1 and Eq. (2.5), the kernel function computes the inner product between documents *p* and *q* in the feature space. For our case, this can be written as:

$$\kappa_{CWK}\ (d_p, d_q) = \langle \bar{\phi}(d_p)\ , \bar{\phi}(d_q)^T \rangle = \phi(d_p)S\ \ S^T \phi(d_q)^T \tag{5.9}$$

where κ (d$_p$, d$_q$) is the similarity value between documents d$_p$ and d$_q$, S is the class-based semantic term relation matrix which makes use of the weights of terms according to Eq. (2.18). In other words, here S is a semantic proximity matrix of terms.

As in [27], for SVM and other kernel-based approaches the information is stored in Gram matrix or kernel matrix which is given by:

$$G_{p,q} = k_{CWK}(d_p, d_q) \tag{5.10}$$

The Gram matrix or kernel matrix are essentially equivalent. By operating on one of these matrixes, it is easy to encode the data in a more appropriate way for mining and learning [27]. Additionally, as it is mentioned in Section 2.1, to be a valid kernel function, the Gram matrix that is formed from the kernel function must satisfy the Mercer's conditions[25]. These conditions are satisfied when the Gram matrix is positive semi-definite. It has been shown in [107] that the matrix *G* formed by the kernel function Eq. (5.9) with the outer matrix product SS^T is indeed a positive semi-definite matrix.

After all calculations we obtain*W* as a term-by-class matrix which includes weights of terms in all classes. We observe that these weights reflect the importance of those words in order to distinguish the classes. Indeed after calculations, terms semantically close to the theme discussed in the documents of that class, gain the highest weight in the range. In other words semantically related terms of that class, i.e. "core" words[3], gain importance while semantically isolated terms, i.e. "general" words, lose their importance. So terms are ranked based on their importance.

Consequently we argue that *S* which is based on *W* performs a similar kind of semantic smoothing as in Eq. (5.9). The new representation of the documents is richer than the standard representation with TF-IDF since; supplementary statistical information is directly calculated from our training corpus and embedded into the kernel function. In other words transformations in Eq. (5.9) smooth the simple term vector representation using semantic ranking while moving from the original input space to a feature space through kernel transformation functions $\bar{\phi}(d_p)$ and $\bar{\phi}(d_q)$ for the documents d_p and d_q,respectively as in. As it is explicitly mentioned in[106], the presence of S in Eq. (5.9), changes the orthogonality of the document vectors, as this mapping introduces term dependencies. Documents can be seen similar even if they do not have any common terms by eliminating orthogonality.

We also observe that the class-based weighting calculation degrades stop words by assigning them very small weights similar to the[94]. Let us consider the following two cases, which are represented in Table 5.9similar to[94]. According to Table 5.9, $t_1$and t_2 are occurred in one or more documents of c_1, not in remaining classes; c_2, c_3 and c_4 while t_3 and $t_4$are occurred in one or more documents of c_3, not in remaining classes; c_1, c_2 and c_4, respectively. In other words $t_1$and t_2 are significant words of the theme discussed in c_1, while t_3 and t_4 are significant terms of the topic discussed in c_3; getting high weight according to Eq. (2.18); since the number of documents that term w occurs in the entire corpus that is N_w is inversely proportional to the weight. Also the total frequencies of $t_1$and t_2 in the documents c_1 and the frequencies of t_3 and t_4 in the documents c_3 are directly proportional to the weights of those terms. According to Eq. (2.18), the more a word occurred in only a specific class the higher it gets a weight and conversely the more a word occurred in all classes the less it gets a weight. This statement can also be represented with Table 5.9, since $t_1$and t_2 occurred in only c_1 and t_3 and t_4 occurred in only c_3 while $t_5$and$t_6$are occurred in every classes of the corpus. It is highly possible that these two words, t_5 andt_6, are in the type of “general” words since they are seen in every class of the corpus.

Table 5.9Term frequencies on different classes [94]

	c_1	**c_2**	**c_3**	**c_4**
t_1	1	0	0	0
t_2	1	0	0	0
t_3	0	0	1	0
t_4	0	0	1	0
t_5	1	1	1	1
t_6	1	1	1	1

The algorithm of CWK is as follows:

```
Module Calculating Class-Based Weights of Words
Input
        Training set
Output
        Weight Matrix $W$ for each word $w$ and class $k$
Local variables
          $tfc_{w,k}$ : total term frequency of word $w$ in the documents of class $k$
          $tf_{w,d}$ : total term frequency of word $w$ in the document $d$
          $N$ : total number of documents in the training set
 $N_w$: vector shows the total number of documents in the training set those contain word $w$
begin
          for each word $w$
                    for each document $d_i$ contains word $w$ in the training set
                         $N_w = N_w + 1$ //Increment the number of documents contain word $w$
                    end for
          end for
          for each word $w$
                    for each document $d_i$ in class $k$
                         $tfc_{w,k} = tfc_{w,k} + tf_{w,d}$ //Increment the class frequency of word $w$
                    end for
                    //Calculate the weight of the word $w$ in class $k$
                    $W_{w,k} = (log(tfc_{w,k}) + 1)x(log(N/N_w))$
          end for
end

Module Training                              //Building semantic smoothing kernel
Input
        Training set, $W_{w,k}$
Output
        $G_{p,q}$ : Gram matrix shows the kernel value between documents $d_p$ and $d_q$
```

Local variables

$S_{i,j}$: Semantic smoothing matrix shows the relatedness between words *i* and *j*

```
begin
    $S_{i,j} = W_{w,k} (W_{w,k})^T$          //Building semantic smoothing matrix
    for each document $d_p$ in the training set
        for each document $d_q$ in the training set
                    //Calculating kernel value between documents $d_p$ and $d_q$
            $G_{p,q} = d_p S S^T d_q$
        end for
    end for
end
```

5.2.2 Experimental Results and Discussion

CWK outperforms our baseline kernel at all training set percentages also producing a statistically significant difference based on Students t-Test results on 20 Newsgroups-Science dataset. This can be observed from Table 5.10. The performance gain is specifically obvious at training set percentages 5%, 10% and 30%. For instance at training set percentages 5%, 10% and 30% the accuracies of CWK are 84.31%, 90.94% and 95.89% while the accuracies of linear kernel are 71.44%, 77.97% and 86.73; respectively. CWK also has better performance than our previous semantic kernels IHOSK, and HOTK at all training set percentages as shown in Table 5.10. Also it should be noted that, CWK is superior to our recent study CMK at training set percentages 5%, 10%, 30%, 50% and 70%. There is another point which deserves attention is that by using only 5% of the training set the performance gain of CWK over linear kernel is 18.02%, which is of great importance since usually it is difficult and expensive to obtain labeled data in real world applications. Additionally according to Table 5.10we can conclude that the performance differences of CWK while passing from one training set percentage to another are compatible with the term coverage ratios at those training set percentages. For instance at training set percentage 10%, term coverage jumps to 82.28% from its previous value at 5% that is 63.99%. Similar behavior can be observed at performance of CWK while going through 5% training set percentage to 10% training set percentage; where it generates the accuracies 84.31% and 90.94%; respectively. This means an accuracy change of 6.63% between 5% and 10% training set percentages.

Additional to CWK, we also built a second-order version of CWK with the name Second-Order Class Weighting Kernel (SO-CWK) with the following equation:

$$\bar{\phi}(d) = \phi(d)\ S\ S \tag{5.11}$$

$$\kappa_{SO-CWK}\left(d_p, d_q\right) = \bar{\phi}\left(d_p\right)\ \bar{\phi}\left(d_q\right)^T = \phi(d_p)\ S\ S\ \left(S\ S\right)^T \phi(d_q)^T \tag{5.12}$$

where $\bar{\phi}(d_p)$ and $\bar{\phi}(d_q)$ are transformation functions of kernel from input space into feature space for the documents $d_p$and d_q,respectively.

We also recorded and compared the total kernel computation time of our previous semantic kernels IHOSK and HOTK and CWK. All the experiments presented here are carried on our experiment framework, Turkuaz, which directly uses WEKA [85], [89] on a computer with two Intel(R) Xeon(R) CPUs at 2.66 GHz with 64 GB of memory. Our semantic kernel's computation time on each dataset is recorded in terms of seconds and they are proportionally converted into time units. According to this conversion, for instance on 20 Newsgroups-Science dataset; IHOSK [92], SO-CWK, HOTK [93] and CWK estimates the following time units in order; 100, 60, 56 and 40, respectively which is shown in Figure 5.4.

In IHOSK [92] there is an iterative similarity calculation between documents and terms, which completes totally in 4 steps including the matrix calculations shown in Eq. (4.10) and (4.11). As discussed in [18], producing the similarity matrix of terms SC_t (n×n) has overall complexity $O(tn^3)$ where t is the number of iterations and n is the number of training terms. Similarly, calculating the similarity matrix of documents SR_t (m×m) has $O(tm^3)$ where m is the number of training documents. Since both matrices need to be computed iteratively the overall complexity is bounded by the matrix multiplications of term and document similarity matrices. In our experiments only two iterations are performed (t=2). On the other hand, HOTK[93]has $O(n^3)$ complexity where n is the number of terms as noted in Eq. (4.16). Although both IHOSK and HOTK is bounded by the complexity of matrix multiplications, the IHOSK involves much more matrix multiplications due to the iterative computation of both term and document similarity matrices and consequently runs much longer than HOTK in practice.

The CWK is also bounded by the complexity of matrix multiplication. However, the matrix is much smaller; (n×c), where n is the number of term and c is the number of classes. In addition, the generation of this term by class matrix is also very fast with O(logn) complexity. These differences make CWK faster than HOTK as can be seen in Figure 5.4. On the other hand the, SO-CWK includes an additional matrix multiplication of a term by term matrix, therefore it runs slightly longer than HOTK as shown in Figure 5.4. Since the complexity of IHOSK is higher than both HOTK and the proposed work of the CWK, it is not practical to apply IHOSK on large datasets.

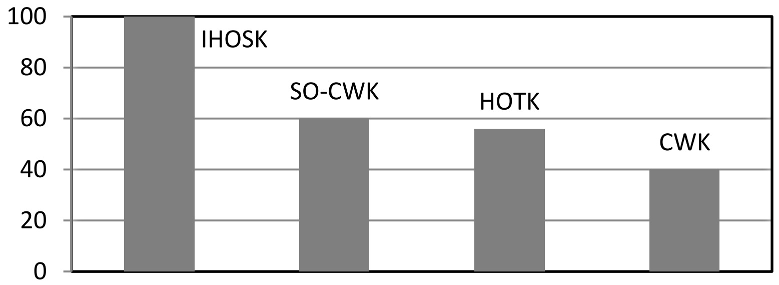


Figure 5.4The total kernel computation time units of IHOSK, SO-CWK, HOTK and CWK on 20 Newsgroups-Science dataset at 30% training set percentage

We also compare CWK with TF-ICF on 20 Newsgroups-Science dataset. The formulation of TF-ICF is given in Eq. (2.16). TF-ICF is a supervised approach as mentioned in Section 2.3. The results are shown in Figure 5.5. According to Figure 5.5, CWK has much better performance than TF-ICF in all training set percentages.

Table 5.10 Accuracy of CWK and other kernels on 20 Newsgroups-Science dataset with varying training set size

TS %	Linear	IHOSK	HOTK	CMK	CWK	Gain	Term Coverage
5	71.44±4.30	84.15±2.87	76.63±2.67	64.51±4.86	**84.31±2.77**	18.02*	63.99
10	77.97±3.73	90.37±0.81	82.47±2.02	82.19±3.58	**90.94±1.72**	16.63*	82.28
30	86.73±1.32	94.31±1.09	89.24±0.74	95.07±0.87	**95.89±0.51**	10.56*	98.01
50	88.94±1.16	94.97±0.90	90.84±1.12	96.71±0.61	**96.82±0.3**	8.86*	99.90
70	90.58±0.93	95.35±0.88	92.06±1.28	**97.12±0.59**	97.08±0.68	7.18*	99.99
80	91.33±1.41	96.23±1.19	93.38±1.43	**97.60±0.66**	97.35±0.56	6.59*	100.00
90	91.40±1.56	96.85±1.70	94.20±1.36	97.75±0.89	**98.20±0.71**	7.44*	100.00

Table 5.11Accuracy of CWK and other kernels on IMDB dataset with varying training set size

TS%	**Linear**	**IHOSK**	**HOTK**	**CWK**	**Gain**	**Term Coverage**
5	76.85±1.31	76.98±1.14	74.21±0.24	**77.62±2.45**	1.00	48.00
10	82.99±1.76	82.55±2.32	82.23±0.42	**84.32±1.19**	1.60	61.51
30	85.57±1.65	87.16±1.64	85.63±1.69	**89.66±0.53**	4.78	86.35
50	88.46±1.89	89.40±1.91	87.20±0.33	**91.48±0.69**	3.41	95.91
70	89.93±1.18	91.31±0.87	90.41±0.55	**92.75±1.05**	3.14	99.17
80	90.65±1.09	92.38±1.43	91.37±0.98	**92.73±1.09**	2.29	99.71
90	91.75±1.14	92.63±1.19	91.59±0.27	**93.60±1.52**	2.02	99.98

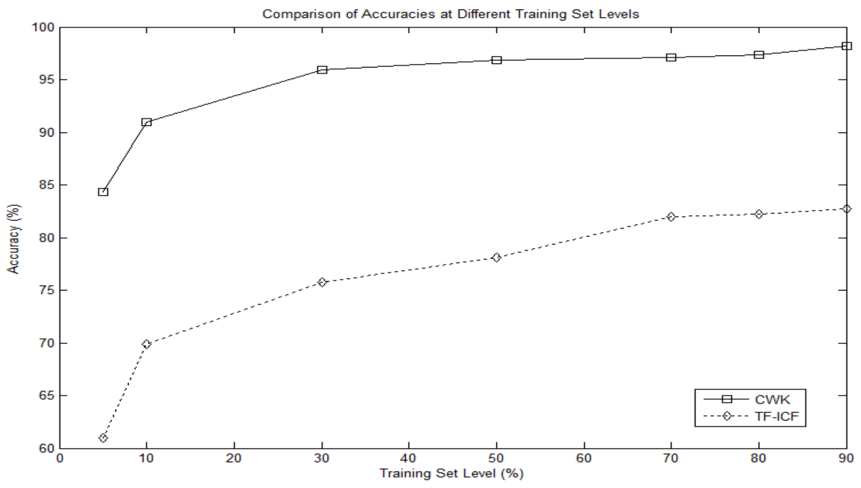


Figure 5.5The Comparison of the accuracies of TF-ICF and CWK at different training set percentages on 20 Newsgroups-Science dataset

According to our experiments, CWK demonstrates a notable performance gain on IMDB dataset, which can be seen in Table 5.11. At 30% training set level, the performance of CWK is 89.66% while the performance of linear kernel is only 85.57%. It is also very promising to see that CWK is superior to both linear kernel and our previous algorithms IHOSK [92] and HOTK [93] throughout all training set percentages.

Table 5.12denotes experiment results on 20 Newsgroups-Politics dataset. In this dataset, again CWK's performance is better than linear kernel's at all training set percentages. Similarly, CWK performs better than both IHOSK and HOTK at all training set percentages. Additionally, CWK statistically significantly outperforms our baseline kernel at 5% training set level based on Students t-Tests.

Table 5.12Accuracy of CWK and other kernels on 20 Newsgroups-Politics dataset with varying training set size

TS%	Linear	IHOSK	HOTK	CWK	Gain	Term Coverage
5	79.01±2.65	82.27±4.60	80.72±1.56	**83.49±4.16**	5.67*	58.60
10	84.69±1.24	88.61±2.10	84.89±2.15	**88.73±2.29**	4.77	75.02
30	92.04±1.06	93.61±1.08	88.31±1.22	**94.90±0.97**	3.11	96.37
50	93.73±0.57	93.55±3.58	90.29±0.79	**96.15±0.80**	2.58	99.43
70	94.55±1.21	93.24±3.08	90.15±1.15	**95.87±0.90**	1.40	99.97
80	94.03±0.91	95.30±1.82	92.50±1.60	**96.80±1.09**	2.95	100.00
90	94.86±1.26	95.80±2.28	92.46±2.01	**96.27±1.97**	1.49	100.00

For 20 Newsgroups-Compdataset, CWK significantly outperforms linear kernel at all training set percentages similar to the situation in 20 Newsgroups-Science dataset as shown in Table 5.13. The highest gain of CWK over linear kernel on this dataset is at 5% training set level with 18.52% gain. Overall, the gains of CWK over linear kernel are usually much larger in 20 Newsgroups-Compdataset compare to the 20 Newsgroups-Politics. This may due to the larger number of classes in 20 Newsgroups-Compdataset.

Experiment results on 20 Newsgroups-Religiondataset are presented in Table 5.14. These results show that CWK has superiority over other kernels starting from 10% training set level. For instance at training set percentage 30% CWK's gain over linear kernel is 7.55%. Also in training set percentage 30% CWK shows a significant improvement over linear kernel.

Table 5.13Accuracy of CWK and other kernels on 20 Newsgroups-Compdataset with varying training set size

TS%	Linear	IHOSK	HOTK	CWK	Gain	Term Coverage
5	56.75±4.72	68.12±1.04	60.22±3.00	**67.26±2.53**	18.52*	48.26
10	65.45±2.77	72.71±0.43	66.70±1.14	**76.60±1.53**	17.04*	65.19
30	75.38±2.12	78.71±0.04	75.97±1.04	**84.67±0.58**	12.32*	91.51
50	77.89±1.60	82.18±1.13	78.68±0.71	**87.09±0.77**	11.81*	98.92
70	79.63±1.59	84.67±2.83	80.97±1.18	**87.63±1.53**	10.05*	99.83
80	79.00±2.25	85.81±0.54	81.58±1.85	**88.10±0.96**	11.52*	99.98
90	81.40±2.47	85.96±0.69	81.32±1.46	**87.88±2.56**	7.96*	100.00

Table 5.15presents the experiment results on Mini-newsgroups dataset. According to these results CWK outputs better accuracy results than linear kernel at all training set percentages. Furthermore, it should be noted that at 5% training set level, the gain of CWK over linear kernel is 21.55% which is a very important advantage on the classification accuracy given the very limited labeled information. On the other hand, at training percentages 50%, 70%, 80% and 90% HOTK generates higher accuracy than CWK. This can be explained with the capability of HOTK to capture latent relations between terms with its higher-order approach[93].

Table 5.14Accuracy of CWK and other kernels on 20 Newsgroups-Religiondataset with varying training set size

TS%	**Linear**	**IHOSK**	**HOTK**	**CWK**	**Gain**	**Term Coverage**
5	74.73±2.47	**77.73±2.47**	65.33±1.70	75.32±2.87	0.79	41.80
10	80.98±2.69	81.19±1.92	72.10±1.95	**82.63±1.97**	2.04	59.03
30	83.87±0.78	84.85±1.84	83.50±1.58	**90.20±1.08**	7.55*	88.18
50	88.39±0.93	88.96±2.30	86.19±1.35	**92.41±0.44**	4.55	96.16
70	89.68±1.41	90.62±1.18	87.26±0.31	**92.62±0.99**	3.28	99.37
80	90.70±1.12	91.00±0.20	88.90±0.24	**93.17±1.21**	2.72	99.80
90	91.65±1.63	91.70±1.73	89.00±2.37	**93.20±1.66**	1.69	99.99

Since some of the datasets used in this study [95] are also used in other studies such as[14], [108] we have the opportunity to compare our results with them. The first algorithm we compare our results is Higher-Order SVM (HOSVM)[14]. For instance at 30% training set level of 20 Newsgroups-Compdataset; 75.38%, 78.71%, 75.97% and 84.67% accuracies are obtained by linear kernel, IHSOK, HOTK and CWK respectively as mentioned above. On the same training set level, HOSVM achieves 78% accuracy according to the Figure. 2(d) in[14].This comparison shows CWK outperforms HOSVM by approximately 8.55% gain. Actually CWK's superiority on HOSVM can also be seen on other datasets such as 20 Newsgroups-Religion, 20 Newsgroups-Science and 20 Newsgroups-Politics. For instance at 30% training set level on 20 Newsgroups-Politics dataset while HOSVM's performance is about 91%, where CWK reaches 94.9% accuracy, which corresponds to 4.29% gain. Very similar picture can be seen at a higher training set level such as 50%. For example, the accuracy values of 88.94%, 92%, 94.97%, 90.84%, 96.82% are achieved by linear kernel, HOSVM, IHSOK, HOTK and CWK respectively on 20 Newsgroups-Science dataset at this level.

Moreover, we also have the chance to compare our results with the study in [108]. Harish et al. proposes a text classification algorithm in [108] which uses B-Tree and preserves the term sequence with a data structure called Status Matrix. One of the datasets they use is Mini-newsgroups dataset. They do not apply any attribute selection technique such as IG in their preprocessing phase. For instance at training setpercentage 50% on Mini-newsgroups dataset; 78.87%, 79.24%, 85.54% and 84.44% accuracies are achieved by linear kernel, IHSOK, HOTK and CWK respectively as

Table 5.15Accuracy of CWK and other kernels on Mini-newsgroups dataset with varying training set size

TS%	**Linear**	**IHOSK**	**HOTK**	**CWK**	**Gain**	**Term Coverage**
5	52.38±5.53	61.29±1.03	49.69±5.64	**63.67±3.31**	21.55*	34.90
10	59.85±3.88	64.15±0.54	66.24±3.81	**71.66±1.22**	19.73*	50.08
30	72.84±3.56	75.51±0.31	81.82±2.04	**81.93±1.13**	12.48*	76.16
50	78.87±2.94	79.24±0.31	**85.54±1.20**	84.44±1.14	7.06*	87.65
70	80.05±1.96	79.73±0.45	**87.28±1.13**	84.85±1.56	6.00*	94.27
80	82.63±1.36	83.05±0.58	**88.15±1.58**	86.43±1.16	4.60	96.22
90	84.65±2.48	85.38±1.28	**88.10±2.80**	85.05±3.93	0.47	98.55

shown in Table 5.15. On the same training set percentage, the maximum accuracy gathered by the study in [108] is 68.95. According to this comparison we observe that CWK outperforms the algorithm in [108] by 15.49%.

CHAPTER 6

INSTANCE LABELING IN SEMI-SUPERVISED LEARNING USINGMEANING VALUES OF TERMS

In supervised learning systems, only labeled samples are used for building a classification model which is then used to predict the class memberships of the unlabeled samples. However, obtaining labeled data is very expensive, time consuming and difficult in real-life practical situations as labeling a data set requires the efforts of human experts. On the other side, unlabeled data are often plentiful which makes it relatively inexpensive and easier to obtain. Semi-Supervised Learning (SSL) methods strive to utilize this abundant source of unlabeled instances to improve the learning capacity of the classifier especially when amount of labeled instances are limited. Since SSL techniques reach higher accuracy and require less human effort, they attract a substantial amount of attention both in practice applications and theoretical research. Although the use of unlabeled data compromises a new opportunity to have a better classification, how to use them to improve prediction is still an open research issue. In this approach, we offer a new semi-supervised algorithm, which utilizes a new method to predict the class labels of unlabeled examples in a corpus and incorporate them into the training set to build a better classifier. The approach presented here depends on a meaning measure, which calculates the meaningfulness of the terms in the context of classes. The meaning measure is based on the Helmholtz principle from Gestalt theory and applied to several text-mining applications such as document summarization and feature extraction. However, to the best of our knowledge, ours is the first study to use meaning measure in a semi-supervised setting to build a semantic kernel for Support Vector Machines (SVM). We evaluated the proposed approach by conducting a large number of experiments on well-known textual datasets and present results with respect to different experimental conditions. We compare our results with standard linear kernel which is the traditional state of the art algorithm in SVM for text classification, semi-supervised form of linear kernel and a supervised classification algorithm which also uses meaning measure, from our previous study. Our results show that labeling unlabeled instances based on meaning values of terms is valuable, and increase the classification accuracy significantly.

6.1Instance Labeling Based on Meaning (ILBOM)

6.1.1 Methodology

ILBOM is mainly composed of five independent modules including preprocessing, meaning calculation, labeling, kernel evaluation and classification. The architecture of ILBOM System is shown in Figure 6.1.

6.1.1.1Meaning Calculation

We use Eq. (8) in order to calculate the meaning values of the terms in this labeled set which produces M_{train} class-based term meaning matrix is made up of *t* rows (terms) and *j* columns (classes). The $M_{labeled}$ matrix shows the meaningfulness of the terms in the labeled set for each class. If a word occurs only once in a class then its meaning value for that class is zero according to Eq. (2.32). If a word does not occur at all in a class, it gets minus infinity based on Eq. (2.32) as a meaning value for that class. In order to make calculations more practical we assign the next smallest value to that word according to the range of meaning values we get for all the words in our labeled documents. After all calculations we get $M_{labeled}$ as a term-by-class matrix which includes the meaning values of terms in all classes of the labeled documents. We observe that these meaning values are high for those words that allow us to distinguish between classes. Indeed terms semantically close to the theme discussed in the class gain the highest meaning values. In other words semantically related terms of that class, i.e. “core” words as mentioned in [3], gain importance while semantically isolated terms, i.e. “general” words lose their importance. Also as it is mentioned in [99] and utilized in [94], meaning calculation automatically filters stop words by assigning them very small amounts of meaning values.

Our approach, ILBOM, utilizes both labeled and unlabeled data. In order to incorporate unlabeled examples into the classifier model in SVM, we calculate the total meaning value of an unlabeled document using Eq. (6.1):

$$TM(d_i, c_j) = \sum_{n=1}^{t} M_{labeled}(w_{nj}) \times tfw_{n,d_i} \qquad (6.1)$$

where $M_{labeled}(w_{nj})$ is the meaning value of the termw_n for the class $c_j$as mentionedabove, tfw_{n,d_i} is the number of occurrence of the term w_n in the document d_i and $TM(d_i, c_j)$ is the total meaning value of the document $d_i$for the class c_j.

6.1.1.2 Labeling

In $TM(d_i, c_j)$ matrix, $d_{ik} = [d_{i1},\ldots,d_{ik}]$ is the document vector showing the document d_i's total meaning values from the aspects of all the classes, respectively. We simply select the column (class number) with the greatest value in $d_{ik} = [d_{i1},\ldots,d_{ik}]$ document vector and label this document with this class number. After completing this labeling-step, all the unlabeled instances are assigned labels and the updated version of the labeled instances are found as follows:

$$L = L_o + L_p \qquad (6.2)$$

where $L_o$is the original labeled instances, L_p is the previously unlabeled instances with their current predicted labels and L is the total of $L_o$and L_p; respectively.

6.1.1.3 Kernel Evaluation

In this step we simply run the CMK which is proposed in our previous study [94] on L. In CMK, we build the class-based term meaning matrix M using meaning calculations given in Eq. (2.32). The M matrix shows the meaningfulness of the terms in each class. Based on M we calculate S matrix in order to reveal class based semantic relations between terms. Specifically, the i, j element of S quantifies the semantic relatedness between terms t_i and t_j.

$$S = M \ M^T \qquad (6.3)$$

The S is a semantic smoothing matrix to transform documents from input space to feature space. Thus, S is a symmetric term-by-term matrix. Mathematically, the kernel value between two documents is given as

$$k_{ILBOM}\ (d_1, d_2) = d_1\ S\ S^T d_2^T \tag{6.4}$$

where k_{ILBOM} *(d_1, d_2)* is the similarity value between documents d_1 and d_2, S is the semantic smoothing matrix.

As mentioned in[106], the presence of *S* in Eq. (6.4) changes the orthogonality of the vector space model, as this mapping introduces term dependence. Documents can be seen as similar even if they do not share any terms by eliminating orthogonality.

6.1.1.4 Classification

We integrated our kernel function into the implementation of the SVM algorithm in WEKA[85]. In other words, we built a kernel function that can be directly used with Platt's Sequential Minimal Optimization (SMO) classifier [86]. In the classification-step all the test instances' labels are predicted and the classification error rate is calculated.

6.1.2Experimental Results and Discussion

In order to compare the results of ILBOM we use two baseline algorithms. First of them is called SSL-Linear. Please note that linear kernel is the traditional state of the art algorithm in SVM for text classification [19], [20]. SSL-Linear first classifies unlabeled examples by using linear kernel that is trained by only the labeled examples. Then, like ILBOM it merges the labeled examples and unlabeled examples with their pre-labels and builds the trainer by using standard linear kernel. After that it again attempts to classify unlabeled examples via the last built model and compares the labels of an instance. If an instance is classified into a different class by the second classifier then its label is updated since the final model is more comprehensive than the first model and is expected to produce predictions with higher classification confidence. The self-training process ends when either there is no change in the predictions or it reached 100 iterations. We also compare the results of ILBOM to those of CMK, which is the second baseline.

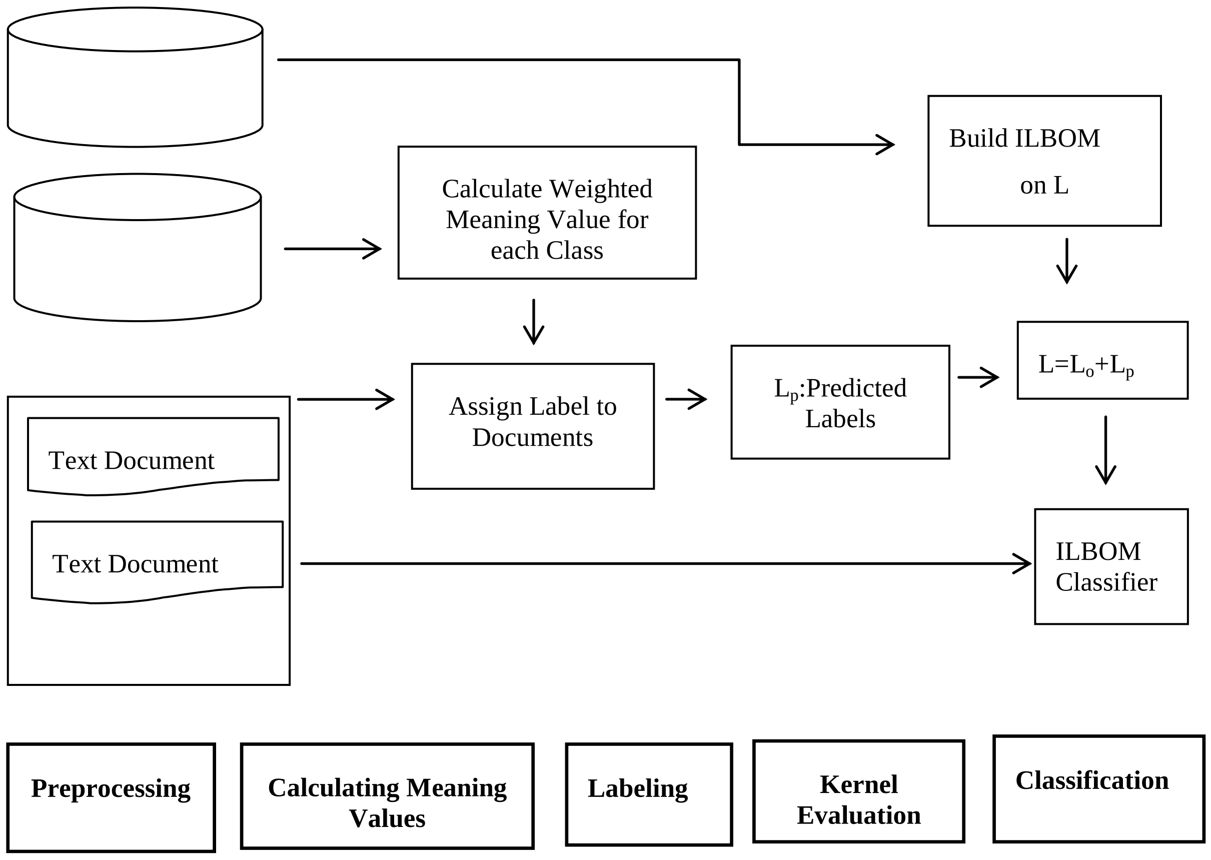


Figure6.1 The architecture of ILBOM system

ILBOM outperforms all of the baseline kernels we used (i.e., SSL-Linear and CMK) which can be observed from Table 6.1, Table 6.2, Table 6.3 and Table 6.4. The performance gain is specifically obvious at smaller labeled set percentages between 1% labeled and 15% labeled set percentage. For instance at labeled set percentages 1%, 2% , 4% , 5% and 7% the accuracies of ILBOM are 59.7%, 64.15%, 79.78%, 86.03% and89.6% while the accuracies of SSL-Linear are 50.03%, 57.1%, 66.45, 67.95% and 71.7%; respectively. ILBOM also has better performance than our previous supervised semantic kernel CMK at most of the labeled set percentages expect labeled set percentages 30% and 50% as shown in Table 6.1, Table 6.2, Table 6.3 and Table 6.4. The highest gain of ILBOM over SSL-Linear kernel on this dataset is at 5% labeled set percentage which is 26.61% that is of great importance since usually it is difficult and expensive to obtain labeled data in real world applications. Also it should be noted that, there are performance gains of ILBOM over linear kernel except labeled set percentage 3%.

Table 6.1, Table 6.2, Table 6.3 and Table 6.4also contain the experiment results on the 20 Newsgroups-Politics dataset. In this dataset, ILBOM's performance is higher than the baseline kernel's in all labeled set percentages except 1% and 50%. Furthermore, ILBOM performs better than both CMK and linear kernel in almost all labeled set percentages except 50%. Only in labeled set percentages 50%, CMK gives better accuracy than ILBOM, but ILBOM still remains better than linear at this labeled set percentage. According to our experiments, ILBOM demonstrates a notable performance gain on the IMDB dataset, which can be seen in Table 6.1, Table 6.2, Table 6.3 and Table 6.4. ILBOM outperforms our baseline, SSL-Linear kernel, in all labeled set percentages also making a significant difference based on Students t-Tests. Table 6.1, Table 6.2, Table 6.3 and Table 6.4 also present the experiment results on Mini-newsgroups dataset. According to these results ILBOM outputs better accuracy than SSL-Linear kernel at all labeled setpercentages except 2% and 50%. But in overall, ILBOM is not as good as supervised kernels (linear kernel and CMK) on this dataset especially between the labeled set percentages 1% and 5%. In other words, interestingly, SSL algorithms (i.e. ILBOM and SSL-Linear) do not benefit from the unlabeled examples on Mini-newsgroups dataset at labeled set percentages under 10%. Because the performance of semi-supervised algorithms (ILBOM and SSL-Linear) at those labeled percentages is less than the performance of linear kernel, which is a supervised algorithm and do not utilize the unlabeled instances. One possible explanation is that ILBOM suffers from capturing enough latent semantics between documents and terms in meaning calculations at low labeled percentages. This may due to the relatively smaller amount of labeled instances per class in this dataset. For instance there are 500 instances per class in 20-newsgroups datasets such as 20 Newsgroups-Science and 1000 instances per class in IMDB but there are only 100 instances per class in Mini-newsgroups. Therefore this yields more misclassified unlabeled examples before building the model and those mislabeled examples degrade the classification performance. Those latent relations may play an important role since the number of classes is relatively high and the number of documents per class is much smaller yielding a higher sparsity.

Table 6.1Accuracy of ILBOM and other kernels on 20 Newsgroups-Science dataset with varying training set size

Labeled %	**Unlabeled %**	**Test %**	**Linear**	**Baseline-1: SSL-Linear**	**Baseline-2: CMK**	**ILBOM**	**Gain**
1	79	20	51.80±5.33	50.03±5.29	39.42±6.78	**59.70±21.63**	19.33*
2	78	20	59.10±5.49	57.10±6.01	50.30±6.00	**64.15±15.2**	12.35*
3	77	20	**66.03±3.61**	64.83±3.64	53.40±7.78	63.93±12.25	-1.39
4	76	20	69.05±3.70	66.45±3.59	60.50±7.17	**79.78±5.62**	20.06*
5	75	20	70.10±4.34	67.95±4.64	70.03±5.07	**86.03±3.77**	26.61*
7	73	20	72.72±4.47	71.70±3.59	78.53±5.07	**89.60±3.01**	24.97*
10	70	20	76.68±2.07	74.58±3.30	87.48±4.81	**92.55±1.23**	24.09*
15	65	20	83.53±2.68	80.53±2.79	89.95±1.71	**94.38±0.91**	17.20*
30	50	20	86.28±2.27	83.70±1.97	**95.28±0.95**	94.98±0.78	13.48*

Table 6.2Accuracy of ILBOM and other kernels on 20 Newsgroups- Politics dataset with varying training set size

Labeled %	**Unlabeled %**	**Test %**	**Linear**	**Baseline-1: SSL-Linear**	**Baseline-2: CMK**	**ILBOM**	**Gain**
1	79	20	52.60±5.69	51.33±6.18	38.60±2.26	**52.63±7.76**	2.53
2	78	20	64.60±6.34	62.20±5.80	47.97±4.64	**82.30±7.78**	32.32*
3	77	20	69.60±5.28	68.97±5.89	64.60±10.43	**86.37±5.43**	25.23*
4	76	20	69.97±5.68	69.07±7.29	68.57±12.7	**88.37±5.05**	27.94*
5	75	20	73.23±3.92	72.23±3.29	78.03±4.46	**88.70±2.8**	22.80*
7	73	20	78.33±5.30	76.87±4.85	80.03±5.24	**92.23±2.26**	19.98*
10	70	20	82.00±2.38	80.77±1.60	87.13±2.13	**93.40±1.28**	15.64*
15	65	20	84.67±4.92	83.93±4.88	91.50±1.69	**94.63±1.62**	12.75*
30	50	20	90.07±1.91	87.50±2.64	94.43±1.05	**95.33±0.82**	8.95*

Table 6.3Accuracy of ILBOM and other kernels on IMDB dataset with varying training set size

Labeled %	**Unlabeled %**	**Test %**	**Linear**	**Baseline-1: SSL-Linear**	**Baseline-2: CMK**	**ILBOM**	**Gain**
1	79	20	65.75±7.79	65.60±8.37	61.33±6.4	**70.43±15.32**	7.36*
2	78	20	69.30±3.28	70.13±2.82	67.97±6.57	**82.88±6.52**	18.18*
3	77	20	72.80±3.28	71.15±3.83	74.25±3.91	**79.25±6.78**	11.38*
4	76	20	76.28±2.27	75.70±2.92	77.03±3.57	**80.08±6.56**	5.79*
5	75	20	77.03±2.55	75.88±2.78	80.33±3.79	**82.68±4.99**	8.96*
7	73	20	79.45±1.76	78.53±2.49	82.38±1.88	**86.33±1.84**	9.93*
10	70	20	79.70±2.86	78.83±3.28	84.65±1.94	**87.08±1.99**	10.47*
15	65	20	81.72±2.47	80.35±1.42	86.98±1.57	**88.70±2.18**	10.39*
30	50	20	85.80±1.37	84.78±1.29	90.88±1.28	**91.40±0.76**	7.81*

Table 6.4Accuracy of ILBOM and other kernels on Mini-Newsgroupsdataset with varying training set size

Labeled %	**Unlabeled %**	**Test %**	**Linear**	**Baseline-1: SSL-Linear**	**Baseline-2: CMK**	**ILBOM**	**Gain**
1	79	20	**36.70±4.24**	29.85±4.04	19.23±2.59	32.45±8.83	8.71*
2	78	20	**38.42±5.47**	30.58±4.61	18.43±2.48	29.23±5.45	-4.41
3	77	20	**46.33±4.32**	38.23±4.71	26.63±3.43	41.90±4.89	9.60*
4	76	20	**46.70±8.34**	38.78±7.55	33.48±4.19	45.63±3.51	17.66*
5	75	20	**50.00±5.49**	40.85±5.63	35.78±3.15	49.75±4.17	21.79*
7	73	20	55.15±4.72	47.70±5.62	47.23±3.18	**59.75±3.88**	25.26*
10	70	20	57.03±2.79	47.98±3.37	53.65±3.27	**64.03±2.16**	33.45*
15	65	20	62.48±4.28	55.48±3.48	61.83±3.24	**67.65±2.85**	21.94*
30	50	20	69.55±4.50	63.60±4.30	70.68±3.38	**72.88±2.4**	14.59*

CHAPTER 7

RESULTS AND DISCUSSION

It has been shown that higher-order co-occurrence relations between documents and terms catch “latent semantics” and result higher accuracies in text classification area [14], [15], [18] and [74]. Motivated by these studies, we propose several corpus-based semantic kernels such as Higher-Order Semantic Kernel (HOSK), Iterative Higher-Order Semantic Kernel (IHOSK) and Higher-Order Term Kernel (HOTK) for SVM. In these studies, we extend the traditional linear kernel (i.e. a dot product between document vectors) for text classification by embedding higher-order relations between terms and documents into the kernel. We show significant improvements on classification performance over linear kernelby taking advantage of higher-order relations between terms and documents. For instance, the HOSK is based on higher-order relations between the documents. The IHOSK is similar to the HOSK since they both propose a semantic kernel for SVM by using higher-order relations. However, IHOSK makes use of the higher-order paths between both the documents and the terms iteratively. Therefore, although, the performance of IHOSK is superior, its complexity is significantly higher than other higher-order kernels. A simplified model, the HOTK, uses higher-order paths between terms.Experiment results show that our higher-order kernels outperform the linear kernel in most of the test cases we tried on several datasets under different training set size conditions. Our results show the usefulness of HOSK, IHOSK and HOTK as semantic kernels for SVM in text classification. As future work, we want to analyze the improved performance of HOSK, IHOSK and HOTK. Especially, we would like to shed light into if and how our approaches implicitly capture semantic information such as synonyms and word sense disambiguation when calculating similarity between documents. Additionally, we plan to get more observations about under what type of conditions HOSK, IHOSK and HOTK perform better than other algorithms.

The other suggested approaches are based on class-based term values. One of them is based on a meaning measure, which we name CMK that calculates the meaningfulness of the terms in the context of classes. Inspiredby the advantages of CMK, we build a semi-supervised algorithm, called ILBOM. The suggested approaches smooth the terms of a document in BOW representation by class-based meaning values of terms. This actually, increases the significance of important or in other words meaningful terms for each class while reducing the importance of general terms which are not useful for discriminating the classes. The meaning values of terms are calculated according to the Helmholtz principle from Gestalt theory in the context of classes.Gestalt theory points out that meaningful features and interesting events appears in large deviations from randomness. The meaning calculations attempt to define meaningfulness of terms in text by using the human perceptual model of the Helmholtz principle from Gestalt Theory. In the context of text mining, the textual data consist of natural structures in the form of sentences, paragraphs, documents, topics and in our case classes of documents. In our semantic kernel setting, we compute meaning values of terms, obtained using the Helmholtz principle in the context of classes where these terms appear. We use these meaning values to smoothen document term vectors. As a result our approach can be considered as a supervised semantic smoothing kernel which makes use of the class information. This is one of the important novelties of our approach since the previous studies of semantic smoothing kernels does not incorporate class specific information. Our experimental results show the promise of the CMK as a semantic smoothing kernel for SVM in the text classification domain. The CMK performs better than linear kernel in most of our experiments. The CMK also outperforms other corpus-based semantic kernels such as IHOSK [92] and HOTK [93], in most of the datasets. Furthermore, the CMK forms a foundation that is open to several improvements. For instance, the CMK can easily be combined with other semantic kernels which smooth the document term vectors using term to term semantic relations, such as the ones using WordNet or Wikipedia.

In our semantic semi-supervised approach, ILBOM, we use these meaning values to smoothen the document term vectors. The main novelty of our approach is to propose a non-iterative yet effective way of assigning labels to unlabeled instances and augment the training set with these in order to build a better performing model using the CMK which is recently proposed in our previous study. It can be considered as a semi-supervised algorithm which is inspired by CMK. Our experimental results show the promise of the ILBOM as a semi-supervised method for SVM in the text classification domain. ILBOM performs better than linear kernel which is commonly used state-of-the art kernel in the literature. We also usetwo baseline algorithms, namely SSL-Linear and CMK, in order to compare the results of ILBOM. According to our experimental results ILBOM outperforms semi-supervised form of linear kernel (SSL-Linear) in most of our experiments. The ILBOM also outperforms the supervised CMK, in most of the datasets. In other words, ILBOM achieves higher classification accuracy by adding unlabeled data into the same amount of labeled data CMK uses. This exciting and convincing result shows that we succeed in building a semi-supervised approach that can benefit from unlabeled data. As future work, we plan to build the self-trained form of our model and analyze the performance differences especially at lower labeled set percentages.

Table 7.1 Accuracy of our semantic kernels and linear kernel on 20 Newsgroups-Science dataset with varying training set size

TS%	Linear	IHOSK	HOTK	CMK	CWK	Term Coverage
5	71.44±4.30	84.15±2.87	76.63±2.67	64.51±4.86	**84.31±2.77**	63.99
10	77.97±3.73	90.37±0.81	82.47±2.02	82.19±3.58	**90.94±1.72**	82.28
30	86.73±1.32	94.31±1.09	89.24±0.74	95.07±0.87	**95.89±0.51**	98.01
50	88.94±1.16	94.97±0.90	90.84±1.12	96.71±0.61	**96.82±0.3**	99.90
70	90.58±0.93	95.35±0.88	92.06±1.28	**97.12±0.59**	97.08±0.68	99.99
80	91.33±1.41	96.23±1.19	93.38±1.43	**97.60±0.66**	97.35±0.56	100.00
90	91.40±1.56	96.85±1.70	94.20±1.36	97.75±0.89	**98.20±0.71**	100.00

We also introduce a new semantic kernel for SVM called Class Weighting Kernel (CWK). CWK is based on weighting the values of terms in the context of classes according to [80] and [81]. CWK smooths the terms of a document in bag-of-words (BOW) representation by making use of the terms' discriminating power for each class in the training set. This in turn increases the importance of significant or in other words,

Table 7.2 Accuracy of our semi-supervised semantic kernel and the other kernels on 20 Newsgroups-Science dataset with varying training set size

Labeled %	**Unlabeled %**	**Test %**	**Linear**	**Baseline-1: SSL-Linear**	**Baseline-2: CMK**	**ILBOM**	**Gain**
1	79	20	51.80±5.33	50.03±5.29	39.42±6.78	**59.70±21.63**	19.33*
2	78	20	59.10±5.49	57.10±6.01	50.30±6.00	**64.15±15.2**	12.35*
3	77	20	**66.03±3.61**	64.83±3.64	53.40±7.78	63.93±12.25	-1.39
4	76	20	69.05±3.70	66.45±3.59	60.50±7.17	**79.78±5.62**	20.06*
5	75	20	70.10±4.34	67.95±4.64	70.03±5.07	**86.03±3.77**	26.61*
7	73	20	72.72±4.47	71.70±3.59	78.53±5.07	**89.60±3.01**	24.97*
10	70	20	76.68±2.07	74.58±3.30	87.48±4.81	**92.55±1.23**	24.09*
15	65	20	83.53±2.68	80.53±2.79	89.95±1.71	**94.38±0.91**	17.20*
30	50	20	86.28±2.27	83.70±1.97	**95.28±0.95**	94.98±0.78	13.48*

core terms for each class while reducing the importance of general terms that exist in all classes. Since this method is used in the transformation phase of a kernel function (from input space into a feature space), it reduces the effects of the several disadvantages of the BOW representation which is discussed in Section 1. We demonstrate that CWK considerably increase the accuracy of SVM compare to the linear kernel by assigning more weight to class specific terms which can be synonymous or very closely related in the context of a class. In other words, CWK uses a class-weighting based semantic smoothing matrix during the transformation from the original space into the feature space of the kernel function. This semantic smoothing mechanism map the similar documents to nearby positions in the feature space of SVM even if they are written using different but semantically closer sets of terms in the context of classes. In our semantic kernel setting, the document term vectors are smoothed based on weights of terms in the context of classes. As a result it can be considered as a supervised semantic kernel which directly makes use of class information. Our experimental results show the promise of CWK as a semantic kernel for SVM in the text classification domain. CWK performs better than linear kernel. The CWK also demonstrates better accuracies than other corpus-based semantic kernels such as IHOSK [92] and HOTK [93], in most of the datasets we used. According to our experimental results, CWK outperforms our baseline kernel at all training set percentages also making a significant difference based on Students t-Tests results on both 20 Newsgroups-Science and 20 Newsgroups-Compdatasets. Furthermore CWK gives higher accuracies than our baseline linear kernel at all of the training set percentages for all of the datasets we used in our experiments. Additionally, on 20 Newsgroups-Science dataset by using only 5% of the training set, the performance gain of CWK over linear kernel is 18.02%, which is of great importance since usually it is difficult and expensive to obtain labeled data in real world applications. A very similar situation is occurred for Mini-newsgroups dataset at again 5% training set level, the performance gain of CWK over linear kernel is 21.55%. Moreover it should be noted that CWK has the capability of reaching more accurate classification performance in compare to both linear kernel and our previous semantic kernels with less execution time than both IHOSK and HOTK. We also believe that CWK forms a foundation that is open to several improvements. The experimental results of these algorithms on 20 Newsgroups-Science dataset are shown in Table 7.1 and Table 7.2. According to experimental results higher-order algorithms have more gain on linear kernel at low training set percentages while corpus-based kernels have more gain on linear kernel at high training set percentages. All the proposed approaches including CWK have independency of the outside semantic sources such as WordNet, so that they can be applied to any language domain. They also form a foundation that can easily be combined with other term-based semantic similarity methods such as unsupervised semantic similarity measures. For instance, the CWK can easily be combined with other semantic kernels which smooth the document term vectors using term to term semantic relations such as the ones using WordNet or Wikipedia. These semantic kernels can also be applied on different domains in TC such as IR, sentimental classification, syntactic-semantic analysis of documents, measuring similarities of short texts such as microblogs, tweets and also can be applied on other domains on otside of TC such as image retrieval, biomedical applications, recommendation systems etc... As future work, we plan to implement different class-based document or term similarities in supervised classification and compare their results to the results of the CWK.

REFERENCES

[1] Salton, G., Yang, C.S., (1973). “On the Specification of Term Values in Automatic Indexing”, Journal of Documentation, 29(4):11-21.

[2] Wang, P., Domeniconi, C., (2008). “Building Semantic Kernels for Text Classification Using Wikipedia”, Proceeding of the 14th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), 713-721.

[3] Steinbach, M. Karypis, G., Kumar, V., (2000). “A Comparison of Document Clustering Techniques”, Proceedings of the KDD Workshop on Text Mining.

[4] Nasir, J. A., Karim, A., Tsatsaronis, G. Varlamis, I., (2011). “A Knowledge-Based Semantic Kernel for Text Classification”, String Processing and Information Retrieval, 261-266, Springer.

[5] Bloehdorn, S., Moschitti, A., (2007). “Combined Syntactic and Semantic Kernels for Text Classification”, Springer, 307-318.

[6] Budanitsky, A., Hirst, G., (2006). “Evaluating WordNet-Based Measures of Lexical Semantic Relatedness”, Journal Comput. Ling. 32(1):13-47.

[7] Lee, J. Ho, Kim, M. H., Lee, Y. J., (1993). “Information Retrieval Based on Conceptual Distance in IS-A Hierarchies”, Journal of Documentation, 49(2):188-207.,

[8] Luo, Q., Chen, E., Xiong, H., (2011). “A Semantic Term Weighting Scheme for Text Categorization”, Journal of Expert Systems with Applications, 38(1):12708-12716.

[9] Nasir, J. A., Varlamis, I., Karim, A., Tsatsaronis, G., (2013). “Semantic Smoothing For Text Clustering”, Knowledge-Based Systems, 54(1): 216-229.

[10] Scott, S., Matwin, S., (1998). “Text Classification Using WordNet Hypernyms”,Proceedings of the ACL Workshop on Usage of WordNet in Natural Language Processing Systems, 45-52.

[11] Siolas, G., d'Alché-Buc, F., (2000). “Support Vector Machines Based On a Semantic Kernel for Text Categorization”, Proceedings of the International Joint Conference on Neural Networks (IJCNN), IEEE,5(1):205-209.

[12] Miller, G., Beckwith, R., Fellbaum, C., Gross, D., Miller K., (1993). “Five Papers on WordNet”, Technical report, Stanford University.

[13] Zhang, P. Y., (2013). “A HowNet-Based Semantic Relatedness Kernel for Text Classification”, Indonesian Journal of Electrical Engineering (TELKOMNIKA) 11(4):1909-1915.

[14] Ganiz, M. C., Lytkin, N. I., & Pottenger, W. M., (2009). “Leveraging Higher-Order Dependencies between Features for Text Classification”, Proceedings of the Conference Machine Learning and Knowledge Discovery in Databases (ECML/PKDD), 375-390.

[15] Ganiz, M.C., George, C., Pottenger, W.M., (2011). “Higher-Order Naive Bayes: A Novel Non-IID Approach to Text Classification”, IEEE Transactions on Knowledge and Data Engineering,23(7):1022-1034.

[16] Poyraz, M., Kilimci, Z.H., Ganiz, M.C., (2012). “A Novel Semantic Smoothing Method Based on Higher-Order Paths for Text Classification”, IEEE International Conference on Data Mining (ICDM), 615-624.

[17] Poyraz, M., Kilimci, Z.H., Ganiz, M.C., (2014). “Higher-Order Smoothing: A Novel Semantic Smoothing Method for Text Classification”, Journal of Computer Science and Technology, 29(3):376-391.

[18] Bisson, G and Hussain, F., (2008). “Chi-Sim: A New Similarity Measure for the Co-Clustering Task,” Proceedings of the Seventh International Conference on Machine Learning and Applications, 211-217.

[19] Joachims, T., (1998). “Text Categorization with Many Relevant Features”, Proceedings of European Conference on Machine Learning, Springer Verlag, 137-142.

[20] Dumais, S., Platt, J., Heckerman, D. and Sahami, M., (1998).“Inductive Learning Algorithms and Representations for Text Categorization”, Proceedings of the Seventh International Conference on Information Retrieval and Knowledge Management (ACM-CIKM-98),148-155.

[21] Zhu, X. J., (2005). “Semi-supervised Learning Literature Survey, Technical Report”,Department of Computer Sciences, University of Wisconsin at Madison, Madison, WI.

[22] Mitchell, T., (1997). “Machine Learning”, McGraw-Hill Computer Science series.

[23] Boser, B. E., Guyon, I. M., Vapnik, V. N., (1992). “A Training Algorithm for Optimal Margin Classifier”, Proceedings of the 5th ACM Workshop, Comput. Learning Theory, 144-152.

[24] Vapnik, V.N., (1995). “The Nature of Statistical Learning Theory”, Springer, New York.

[25] Alpaydın, E.,(2004). “Introduction to Machine Learning”, MIT press.

[26] Hsu, C.W., Lin, C.J., (2002). “A Comparison of Methods for Multiclass Support Vector Machines”, IEEE Transactions on Neural Networks 13(2): 415-425.

[27] Wang, T., Rao, J., Hu, Q., (2014). “Supervised Word Sense Disambiguation Using Semantic Diffusion Kernel”, Engineering Applications of Artificial Intelligence, Elsevier, 27(1):167-174.

[28] Anthony, G., Gregg, H. and Tshilidzi, M., (2007). “Image Classification Using SVMs: one-against-one vs. one-against-all,” In Proceedings of the 28th Asian Conference on Remote Sensing.

[29] Pal, M., (2008). “Multiclass Approaches for Support Vector Machine Based Land Cover Classification”, Proceedings of the 8th Annual International Conference, Map India.

[30] Melgani, F., and Bruzzone, L., (2004). “Classification of Hyperspectral Remote Sensing Images with Support Vector Machines”, IEEE Transactions on Geoscience and Remote Sensing, 42(1):1778-1790.

[31] Hastie, T. J. and Tibshirani, R. J., (1998). “Classification by Pairwise Coupling”, Advances in Neural Information Processing Systems, The MIT Press, 26(2):451-471.

[32] Knerr, S., Personnaz, L., and Dreyfus, G., (1990). “Single-Layer Learning Revisited: A Stepwise Procedure for Building and Training Neural Network”, Neurocomputing: Algorithms, Architectures and Applications, NATO ASI, Berlin: Springer-Verlag, 41-50.

[33] Moore, A., (2003). “Support Vector Machines”, Tutorial slides, http://www.cs.cmu.edu/~awm.

[34] Cristianini, N. and Shawe-Taylor, J., (2000). “An Introduction to Support Vector Machines and Other Kernel-based Learning Methods”, Cambridge University Press.

[35] Burges C., (1998). “A Tutorial on Support Vector Machines for Pattern Recognition”, In “Data Mining and Knowledge Discovery”, Kluwer Academic Publishers, Boston, 2(2):121-167.

[36] Howley, T. and Madden, M. G., (2005). “The Genetic Kernel Support Vector Machine: Description and Evaluation,” Artif. Intell. Rev., 24(3):379-395.

[37] Reborto de Souza, C., (2012).“Kernel Functions for Machine Learning Applications”, http://crsouza.com.

[38] Lewis, J.P., (2004).“Tutorial on SVM”, CGIT Lab, USC.

[39] Kandola, J., Shawe-Taylor, J., Cristianini, N., (2004). “Learning Semantic Similarity”, Advances in Neural Information Processing Systems 15(1):657–664.

[40] Tsatsaronis, G., Varlamis, I. Vazirgiannis, M., (2010). “Text Relatedness Based on a Word Thesaurus”, Journal of Artificial Intelligence Research 37(1):1-39.

[41] Bloehdorn, S., Basili, R., Cammisa, M., Moschitti, A., (2006). “Semantic Kernels for Text Classification Based on Topological Measures of Feature Similarity”, Proceedings of The Sixth International Conference on Data Mining (ICDM), 808–812.

[42] Hotho, A. et al., (2003). “Ontologies Improve Text Document Clustering”, Proceedings of the 3rd IEEE International Conference on Data Mining, 541-544.

[43] Rodriguez, M.B. et al., (2000). “Using WordNet to Complement Training Information in Text Categorization”, Proceedings of 2nd International Conference on Recent Advances in Natural Language Processing II: Selected Papers from RANLP’97, 189 of Current Issues in Linguistic Theory (CILT), 353-364.

[44] Zhang, Z., Gentile, A. L., Ciravegna, F., (2012). “Recent Advances in Methods of Lexical Semantic Relatedness–A Survey”, Natural Language Engineering, 1(1), 1-69.

[45] Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K.., Harshman, R., (1990). “Indexing by Latent Semantic Analysis. J. of the American Society for Information Science”, 41(6): 391-407.

[46] Cavnar, W.B., Trenkle, J.M., (1994). “N-Gram Based Text Categorization”, Proceedings of 3rd Annual Symposium on Document Analysis and Information Retrieval, 161-169.

[47] Fung, B.C.M. et al., (2003). “Hierarchical Document Clustering Using Frequent Itemsets”, Proceedings of SIAM International Conference on Data Mining, 59-70.

[48] Ho, T.B., Funakoshi, K., (1998). “Information Retrieval Using Rough Sets”, Journal of the Japanese Society for Artificial Intelligence 13(3):424-433.

[49] Ho, T.B., Nguyen, N.B., (2000). “Non-hierarchical Document Clustering Based on a Tolerance Rough Set Model”, International Journal of Intelligent Systems 17(1):199-212.

[50] Papka, R., Allan, J., (1998). “Document Classification Using Multiword Features”, Proceedings of the Seventh International Conference on Information and Knowledge Management Table of Contents, Bethesda, Maryland, United States, 124–131.

[51] Lewis, D.D., (1992). “An Evaluation of Phrasal and Clustered Representation on a Text Categorization Task”,SIGIR’92: Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 37-50.

[52] Zhang, W., Yoshida, T., Tang, X., (2008). “Text Classification Based on Multi-Word with Support Vector Machine”, Knowledge-Based Systems, 21(8):879-886.

[53] Blum, A. and Mitchell, T., (1998). “Combining Labeled and Unlabeled Data with Co-Training”, Proceedingsof Conference on Computational Learning Theory, 92-100.

[54] Yarowsky, D., (1995). “Unsupervised Word Sense Disambiguation Rivaling Supervised Methods”, Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics, 189-196.

[55] Rosenberg, C. et al., (2005). “Semi-Supervised Self-Training of Object Detection Models”, Seventh IEEE Workshop on Applications of Computer Vision.

[56] Zhou, D. et al., (2004). “Semi-Supervised Learning on Directed Graphs”, Advances in Neural Information Processing Systems, 1633-1640.

[57] Nigam, K. and R. Ghani, (2000). “Analyzing the Effectiveness and Applicability of Co-Training”, Proceedings of the 9th ACM International Conference on Information and Knowledge Management, Washington, DC, 86-93.

[58] Chapelle, O. and Zien, A., (2005). “Semi-Supervised Classification by Low Density Separation”, Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, 57-64.

[59] Kasabov, N. and Pang, S., (2004). “Transductive Support Vector Machines and Applications in Bioinformatics for Promoter Recognition”, Neural Information Processing-Letters and Reviews, 3(2):31-38.

[60] Goldman, S. and Zhou, Y., (2000). “Enhancing Supervised Learning with Unlabeled Sata”, Proceedings of the 17th ICML, San Francisco, CA, Morgan Kaufmann, 327-334.

[61] Jin, Y., Huang, C., & Zhao, L., (2011). “A Semi-Supervised Learning Algorithm Based on Modified Self-training SVM”, Journal of Computers, 6(7):1438-1443.

[62] Nigam, K. et al., (2000). “Text Classification from Labeled and Unlabeled Documents Using EM”, Machine Learning, 39(2):103-134.

[63] Muslea, I., Minton, S., Knoblock, C.A., (2002). “Active Semi-Supervised Learning in Robust Multi-view Learning”, Proceedings of the Nineteenth International Conference on Machine Learning, 435-442.

[64] Liu, A., Jun, G., Ghosh, J., (2009). “A Self-Training Approach to Cost Sensitive Uncertainty Sampling”, Machine Learning 76:257-270.

[65] Chapelle, O., Scholkopf, B., Zien, A. (2006). “Semi-Supervised Learning”, MIT Press, Cambridge.

[66] Wang, B., Spencer, B., Ling, C.X., Zhang, H., (2008). “Semi-Supervised Self-Training for Sentence Subjectivity Classification”, The 21st Canadian Conference on Artificial Intelligence, 344–355.

[67] Li, M., & Zhou, Z. H., (2005). “SETRED: Self-Training with Editing”, Advances in Knowledge Discovery and Data Mining, Springer Berlin Heidelberg, 611-621.

[68] Guo, Y., Zhang, H., Liu, X., (2011). “Instance Selection in Semi-Supervised Learning”,Proceedings of 24th Canadian Conference on Artificial Intelligence, 158-169.

[69] Li, K., Zhang, W., Ma, X., Cao, Z., & Zhang, C.,(2008), “A Novel Semi-Supervised SVM Based on Tri-training”, Intelligent Information Technology Application, IITA'08. Second International Symposium on 3(1):47-51.

[70] Cozman, F.G. et al., (2003). “Semi-Supervised Learning of Mixture Models”, Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), 99-106.

[71] Guo, Y., Niu, X., Zhang, H., (2010). “An Extensive Empirical Study on Semi-supervised Learning”, the 10th IEEE International Conference on Data Mining, 186-195.

[72] Li, Y.F., Kwok, J.T., Zhou, Z.H., (2010). “Cost-Sensitive Semi-Supervised Support Vector Machine” Proceedings of 24th AAAI Conference on Artificial Intelligence, 500-505.

[73] Cohen, I., Cozman, F.G., Sebe, N., Cirelo, M.C., Huang, T.S., (2004). “Semi Supervised Learning of Classier: Theory, Algorithms, and Their Application to Human-Computer Interaction”, IEEE Transactions on Pattern Analysis and Machine Intelligence 26(1):1553-1567.

[74] Kontostathis, A., Pottenger, W.M., (2006). “A Framework for Understanding LSI Performance”, Journal of Information Processing & Management, 12(1):56-73.

[75] Jones, K. S., (1972). “A Statistical Interpretation of Term Specificity and Its Application in Retrieval”, Journal of documentation 28(1): 11-21.

[76] Salton, G., and Buckley, C., (1988). “Term-Weighting Approaches in Automatic Text Retrieval”, Inf. Process. Manage, 24(5):513-523.

[77] Debole, F., Sebastiani, F., (2003). “Supervised Term Weighting for Automated Text Categorization”, SAC ’03, Proceedings of the 2003 ACM Symposium on Applied Computing, New York, NY, USA, ACM Press, 784-788.

[78] Deng, Z.-H., Tang, S.-W., Yang, D.-Q., Zhang, M., Li, L.-Y., Xie, K. Q., (2004). “A Comparative Study on Feature Weight in Text Categorization”, APWeb, 3007. Springer-Verlag Heidelberg, 588-597.

[79] Lan, M., Tan, C. L., Su, J., & Lu, Y., (2009). “Supervised and Traditional Term Weighting Methods for Automatic Text Categorization”, Pattern Analysis and Machine Intelligence, IEEE Transactions on, 31(4):721-735.

[80] Biricik, G., Diri, B., Sonmez, A. C., (2009). “A New Method for Attribute Extraction with Application on Text Classification”, Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control (ICSCCW), Fifth International Conference on IEEE, 1-4.

[81] Biricik, G., Diri, B., Sonmez, A. C., (2012). “Abstract Feature Extraction for Text Classification”, Turkish Journal of Electrical Engineering & Computer Sciences, 20(1):1137-1159.

[82] Ko, Y., Seo, J., (2000). “Automatic Text Categorization by Unsupervised Learning”, Proceedings of the 18th conference on Computational Linguistics, Association for Computational Linguistics, 453-59.

[83] Lertnattee, V., Theeramunkong, T., (2004). “Analysis of Inverse Class Frequency in Centroid-Based Text Classification”, IEEE International Symposium on Communications and Information Technology, (ISCIT), 2.

[84] Robertson, S. E. (2004). “Understanding Inverse Document Frequency: On Theoretical Arguments for IDF”, Journal of Document. 60(5):503-520.

[85] Hall, M, Frank, E. Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H., (2009). “The WEKA Data Mining Software: An Update”; SIGKDD Explorations, 11(1).

[86] Platt, J. C., (1998). “Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines”, In Advances in Kernel Method: Support Vector Learning, MIT Press, 185–208.

[87] Amasyali, M.F. and Beken, A., (2009). “Türkçe Kelimelerin Anlamsal Benzerliklerinin Ölçülmesi ve Metin Sınıflandırmada Kullanılması”,IEEE Sinyal İşleme ve İletişim Uygulamaları Kurultayı (SIU), IEEE Press.

[88] Two Text Learning Datasets Website,http://www.cs.cmu.edu/~textlearning, 2013.

[89] Kamber, I.H., Frank, E., (2005). “Data Mining: Practical Machine Learning Tools and Techniques”, 2nd Edition, Morgan Kaufmann, San Francisco.

[90] Han, J., Kamber, M., Pei, J., (2012). “Data Mining: Concepts and Techniques”, Morgan Kaufmann, Third Edition.

[91] Dumais, S, (1993). “LSI meets TREC: A status report”, In Hartman, D., ed., the first Text Retrieval Conference: NIST special publication 105–116.

[92] Altınel, B., Ganiz, M.C., Diri, B., (2014). “A Semantic Kernel for Text Classification Based on Iterative Higher–Order Relations between Words and Documents”, Proceedings of the 13th International Conference on Artificial Intelligence and Soft Computing (ICAISC) , Lecture Notes in Artificial Intelligence(LNAI),8467:505-517.

[93] Altınel, B., Ganiz, M.C., Diri, B., (2014). “A Simple Semantic Kernel Approach for SVM Using Higher-Order Paths”, Proceedings of IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA), 431-435.

[94] Altınel, B., Ganiz, M.C., Diri, B.,(2015). “A Corpus-Based Semantic Kernel for Text Classification by Using Meaning Values of Terms”,Elsevier, Engineering Applications of Artificial Intelligence, August 2015, 43(1): 54–66.

[95] Altınel, B., Diri, B., Ganiz, M.C., (2015). “A Novel Semantic Smoothing Kernel for Text Classification with Class-based Weighting”, Knowledge-Based Systems, August 2015, 89(1):54–66.

[96] Altınel, B., Ganiz, M.C., Diri, B., (2015). “Instance Labeling in Semi-Supervised Learning Using Meaning Values of Terms”,Pattern Recognition Letters ,(Submitted).

[97] Altınel, B., Ganiz, M.C., Diri, B., (2013). “A Novel Higher-Order Semantic Kernel”, ICECCO 2013 (The 10th International Conference on Electronics Computer and Computation), November 7-9, Ankara, Turkey.

[98] Rennie, J. D. M., Shih, L., Teevan, J. and Karger, D. R., (2003).“Tackling the Poor Assumptions of Naive Bayes Classifiers”, Proceedings of International Conference on Machine Learning, 616–623.

[99] Balinsky, A., Balinsky, H., Simske, S., (2010). “On the Helmholtz Principle for Documents Processing”, Proceedings of the 10th ACM Document Engineering (DocEng), 283-286.

[100] Balinsky, A., Balinsky, H., Simske, S., (2011). “On the Helmholtz Principle for Data Mining”, Proceedings of Conference on Knowledge Discovery, Chengdu, China.

[101] Balinsky, A., Balinsky, H., Simske, S., (2011). “Rapid change Detection and Text Mining”, Proceedings of 2nd Conference on Mathematics in Defence (IMA), Defence Academy, UK.

[102] Balinsky, H., Balinsky, A., Simske, S., (2011). “Document Sentences As A Small World”, Proceedings of IEEE International Conference on Systems, Man and Cybernetics (SMC), 2583-2588.

[103] Basili, R., Cammisa, M., Moschitti, A., (2005). “A Semantic Kernel to Classify Texts with Very Few Training Examples”, Proceedings of Workshop Learning in Web Search, 22nd International Conference on Machine Learning (ICML).

[104] Mavroeidis, D., Tsatsaronis, G., Vazirgiannis, M., Theobald, M., & Weikum, G., (2005). “Word Sense Disambiguation for Exploiting Hierarchical Thesauri in Text Classification”, Knowledge Discovery in Databases: PKDD, 181-192, Springer Berlin Heidelberg.

[105] Dadachev, B., Balinsky, A. Balinsky, H.; Simske, S. (2012). “On the Helmholtz Principle for Data Mining”,International Conference on Emerging Security Technologies (EST), 99-102.

[106] Wittek P., Tan, C., (2009). “A Kernel-Based Feature Weighting for Text Classification”, Proceedings of IJCNN-09, IEEE International Joint Conference on Neural Networks, 3373–3379.

[107] Cristianini, N., Shawe-Taylor, J., & Lodhi, H., (2002). “Latent Semantic Kernels”, Journal of Intelligent Information Systems, 18(2):127-152.

[108] Harish, B. S., S. Manjunath, D. S. Guru, (2012).“Text Document Classification: An Approach Based On Indexing”, International Journal of Data Mining & Knowledge Management Process (IJDKP) 2(1): 43-62.

[109] Desolneux, A., Moisan, L., Morel, J.-M., (2008). “From Gestalt Theory to Image Analysis: A Probabilistic Approach”, Interdisciplinary Applied Mathematics, 34, Springer.

[110] Kleinberg, J.,(2002). “Bursty and Hierarchical Structure in Streams”, Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), 7(4):373-397.

[111] Luo, Z.P., Zhang, X.-M., (2008). “A Semi-Supervised Learning Based Relevance Feedback Algorithm in Content-Based Image Retrieval”, Chinese Conference on Pattern Recognition (CCPR ’08),1-4.

[112] Jiang, E.P., (2009). “Semi-Supervised Text Classification Using RBF Networks”, Adams, N.M., Robardet, C., Siebes, A., Boulicaut, J.-F. (Eds.), IDA, Springer, 5772, 95-106.

[113] Li, Y. F. and Zhou, Z. H., (2011). “Towards Making Unlabeled Data Never Hurt”, Pattern Analysis and Machine Intelligence, IEEE Transactions on, 37(1):175-188.

[114] Schwenker, F. and Trentin, E., (2014). “Partially Supervised Learning for Pattern Recognition”, Pattern Recognition Letters, 37(1):1-3.

CURRICULUM VITAE

PERSONAL INFORMATION

Name Surname	:Berna ALTINEL
Foreign Languages	: English
Date of birth and place	: 01.01.1981 / İstanbul
E-mail	:berna.altinel@marmara.edu.tr

EDUCATION

Degree	**Field**	**School/University**	**Year of Graduation**
Master	Computer Engineering	Istanbul Technical University	2007
Undergraduate	Computer Engineering	Yeditepe University (with %100 Scholarship)	2004
High School	Science and Maths	Haydarpaşa Anatolia High School,Istanbul	1999

WORKING EXPERIENCES

Year	**Firm/Company**	**Role**
2009-2012	TÜBİTAK	Test Engineer
2006-2009	SoftTech Software Company (Participant of İşBank)	Software Engineer

2005-2006	Argela Software Company	Software Engineer
2004-2005	Troysis Software Company	Software& Test Engineer

PUBLICATIONS

Journal Papers

1. Altınel, B., Ganiz, M.C., Diri, B., (2015). “Instance Labeling in Semi-Supervised Learning using Meaning Values of Terms”, Pattern Recognition Letters,(Submitted).
2. Altınel, B., Ganiz, M.C., Diri, B., (2015). “A Corpus-Based Semantic Kernel for Text Classification by using Meaning Values of Terms”, Elsevier, Engineering Applications of Artificial Intelligence, August 2015, 43(1):54–66.
3. Altınel, B., Diri, B., Ganiz, M.C., (2015). “A Novel Semantic Smoothing Kernel for Text Classification with Class-Based Weighting”,Knowledge-Based Systems, 89(1):265-177.

Conference Papers

1. Altınel, B., Ganiz, M.C., Diri, B., (2014). “A Semantic Kernel for Text Classification Based on Iterative Higher–Order Relations between Words and Documents”, Proceedings of the 13th International Conference on Artificial Intelligence and Soft Computing (ICAISC) , Lecture Notes in Artificial Intelligence(LNAI),8467,505–517.
2. Altınel, B., Ganiz, M.C., Diri, B., (2014). “A Simple Semantic Kernel Approach for SVM Using Higher-Order Paths”, Proceedings of International Symposium on INnovations in Intelligent SysTems and Applications (INISTA), 431-435.
3. Altınel, B., Ganiz, M.C., Diri, B., (2013). “A Novel Higher-order Semantic Kernel”, ICECCO 2013 (The 10th International Conference on Electronics Computer and Computation), November 7-9, Ankara, Turkey.
4. Çataltepe, Z. and Altınel, B., (2009). "Music Recommendation by Modeling User’s Preferred Perspectives of Content, Singer/Genre and Popularity" in "Collaborative and Social Information Retrieval and Access: Techniques for Improved User Modeling" edited by M. Chevalier, C. Julien and C. Soulé-Dupuy, IGI Global, pp. 203-221, ISBN: 978-1-60566-306-7.
5. Çataltepe, Z. and Altınel, B., (2007). "Hybrid Music Recommendation Based on Different Dimensions of Audio Content and Entropy Measure", Eusipco (European Signal Processing Conference), 3-7 September, Poland.
6. Çataltepe, Z. and Altınel, B., (2007). "Music Recommendation Based on Adaptive Feature and User Grouping", ISCIS, Ankara, Turkey.

Research Projects

1. TÜBİTAK-3501, Metinsel Veri Madenciliği için Anlamsal Yarı-eğitimli Algoritmaların Geliştirilmesi,111E239 (2012-2015, Position: Research Assistant).

AWARDS

1. Publication Award from TÜBİTAK-ULAKBİM for: Altınel, B., Ganiz, M.C., Diri, B., (2015). “A Corpus-Based Semantic Kernel for Text Classification by using Meaning Values of Terms”, Elsevier, Engineering Applications of Artificial Intelligence, August 2015, 43(1):54–66.
2. Publication Award from TÜBİTAK-ULAKBİM for: Altınel, B., Diri, B., Ganiz, M.C., (2015). “A Novel Semantic Smoothing Kernel for Text Classification with Class-Based Weighting”, Knowledge-Based Systems, 89(1):265-177.
3. Best Paper of Machine Learning Track at ICECCO 2013 (The 10th International Conference on Electronics Computer and Computation), November 7-9, Ankara, Turkey (2013).
4. Graduation with Honor degree from Department of Computer Engineering, Istanbul Technical University (2007).
5. Graduation with High-Honor degree from Department of Computer Engineering, Yeditepe University (2004).
6. Full scholarship from Yeditepe University, Istanbul, Turkey (1999-2004).
7. Scholarship from Vehbi Koç Foundation (1999-2004).
8. First of the schoolin graduation of Göztepe Secondary School (1995).