MARMARA UNIVERSITY

INSTITUTE FOR GRADUATE STUDIES

IN PURE AND APPLIED SCIENCES

SEMI-SUPERVISED CLASSIFICATION OF DOCUMENTS VIA LABEL PROPAGATION IN GRAPHS

GÖKHAN KOCAMAN

MASTER THESIS

Department of Computer Engineering

Thesis Supervisor

Assoc. Prof. Dr. Murat Can GANİZ

ISTANBUL, 2019

MARMARA UNIVERSITY

INSTITUTE FOR GRADUATE STUDIES

IN PURE AND APPLIED SCIENCES

SEMI-SUPERVISED CLASSIFICATION OF DOCUMENTS VIA LABEL PROPAGATION IN GRAPHS

GÖKHAN KOCAMAN

(524114003)

MASTER THESIS

Department of Computer Engineering

Thesis Supervisor

Assoc. Prof. Dr. Murat Can GANİZ

ISTANBUL, 2019

MARMARA UNIVERSITY

INSTITUTE FOR GRADUATE STUDIES

IN PURE AND APPLIED SCIENCES

Gökhan KOCAMAN, a Master of Science student of Marmara University Institute for Graduate Studies in Pure and Applied Sciences, defended his thesis entitled "**Semi-Supervised Classification of Documents via Label Propagation in Graphs**", on 24.07.2019 and has been found to be satisfactory by the jury members.

Jury Members

Assoc. Prof. Dr. Murat Can GANİZ (Advisor)

Marmara University .....(SIGN).....

Prof. Dr. Banu DİRİ (Jury Member)

Yıldız Teknik Üniversitesi .....(SIGN).....

Assoc. Prof. Dr. Ali Fuat ALKAYA (Jury Member)

Marmara Üniversitesi .....(SIGN).....

APPROVAL

Marmara University Institute for Graduate Studies in Pure and Applied Sciences Executive Committee approves that Gökhan KOCAMAN be granted the degree of Master of Science in Department of Computer Engineering, Computer Engineering Program on 4.07.2019. (Resolution no: 2019/17-07).

Director of the Institute

Prof. Dr. Bülent EKİCİ

Table of Contents

ACKNOWLEDGMENT ii
ÖZET iii
THESIS ABSTRACT iv
LIST OF SYMBOLS v
LIST OF ABBREVIATIONS vi
LIST OF FIGURES vii
LIST OF TABLES viii
1. INTRODUCTION 1
2. BACKGROUND 3
2.1. Word Sense Disambiguation 3
2.2. Label Propagation Algorithm 3
3. RELATED WORK 6
3.1. Approaches for WSD 6
3.2. Label Propagation Algorithms 7
4. APPROACH 11
5. EXPERIMENTS AND RESULTS 12
5.1. Weka vs. Scikit Learn 12
5.2. Gaussian Random Fields (GRF) 12
5.2.1. GRF Parameters 13
5.2.2. Similarity Methods 13
5.2.3. Vectorizer 13
5.2.4. Iteration Count 13
5.2.5. Converge Limit 13
5.3. Label Propagation 13
5.4. Label Propagation vs. GRF 14
5.5. GRF vs. Label Propagation with WSD Data 19
5.6. DILP Algorithm Experiments and Results 21
5.6.1. WSD Datasets 22
5.7. Experimental Setup 38
5.8. Evaluation Results and Discussion 39
6. CONCLUSIONS AND FUTURE DIRECTIONS 41
REFERENCES 42

ACKNOWLEDGMENT

This work is assisted in part by The Scientific and Technological Research Council of Turkey (TÜBİTAK) grant number 116E047 and 118E315. Only the authors are responsible for the points of view in this document and the official position or policies of the TÜBİTAK is not necessarily represented.

I would like to thank my advisor Dr. Murat Can Ganiz of the Department of Computer Engineering at Marmara University, Dr. Bilge Şipal of the Department of Mathematics and Computer Science at Kültür University, and Dr. Berna Altınel of the Department of Computer Engineering at Marmara University. I have always feel myself relax because of his understanding and gentle behavior. When I was in trouble or made a mistake he consistently was patient and did not let me give up.

My wife Sılam was also eager to motivate me when I was in trouble. During working on my thesis she has always tried to be with me and talked me about subjects that I struggled with. Her motivation efforts were really impressive.

07.2019 **Gökhan KOCAMAN**

ÖZET

ÇİZGELERDE ETİKET YAYILIMI İLE BELGELERİN YARI EĞİTİMLİ SINIFLANDIRILMASI

Doğal dil işleme alanının genel sorunlarından biri de kelimenin cümle içerisinde hangi anlamda kullanıldığının tespitidir. Makine öğrenmesi tabiri ile bu bir gözetimlisınıflandırma sorunudur. Buna alternatif olarak eğer elimizde çok az etiketli veri bulunuyorsa ve çok sayıda etiketsiz veri varsa yarı gözetimli sınıflandırıcılar kullanılabilir. Biz bu çalışmamızda kelime anlam ayrımı için en bilinen sınıflandırma algoritmalarından biri olan Label Propagation algoritmasını iyileştirmeyi amaçladık. Yaklaşımımız ise semantik diffusion kernel kullanmak olacaktır. Geliştirdiğimiz bu algoritmayı diffused label propagation algorithm (DILP) olarak adlandırdık. Algoritmamızı belirginleştirilmiş yapıda bulunan farklı boyutlardaki eğitim kümeleri ile denedik. Yaptığımız bu deneyler ile şu sorulara cevap vermeye çalıştık: 1. Ürettiğimiz semantic kernel formülü mevcut popüler kernellerden daha iyi performans sağlıyor mu? 2. Hangi koşullarda diğer kernellerden daha iyi sonuç vermektedir? 3. Ne tür regülasyon metodu daha iyi sonuç verir? Deneylerimiz sonucunda bazı koşullarda yaklaşımımızın daha iyi performans gösterdiği gözlemlenmiştir.

THESIS ABSTRACT

SEMI-SUPERVISED CLASSIFICATION OF DOCUMENTS VIA LABEL PROPAGATION IN GRAPHS

Word sense disambiguation is the one of major natural language processing problems. It is used to identify the correct sense of a many sense word based on its context. In terms of machine learning, this can be considered as a supervised classification problem. A better alternative can be the use of semi-supervised classifiers since labeled data is usually scarce yet we can access large quantities of unlabeled textual data. We propose an improvement to Label Propagation which is a well-known transductive classification algorithm for word sense disambiguation. Our approach make use of a semantic diffusion kernel. We name this new algorithm as Diffused Label Propagation Algorithm (DILP). We evaluate our proposed algorithm with experiments utilizing various sizes of training sets of disambiguated corpora. Our experiments help us to answer the following questions: 1. Does our algorithm with semantic kernel formulation yield higher classification performance than the popular kernels? 2. Under which conditions does a kernel design perform better than others? 3. What kind of regularization methods result with better performance? Our experiments demonstrate that our approach can outperform baseline in terms of accuracy in several conditions.

LIST OF SYMBOLS

L Graph Laplacian

G term x term matrix

Y_L Known sense

Y_u Unknown sense

X Term x Document Matrix

Λ Control parameter

LIST OF ABBREVIATIONS

Bi-LSTM	Bidirectional Long Short-Term Memory
DILP	Diffused Label Propagation algorithm
GRF	Gaussian Random Field
IMS	It Makes Sense
KNN	K-Nearest Neighbor
LP	Label Propagation
ML	Machine Learning
POS	Part of Speech
RBF	Radial Basis Function
SVM	Support Vector Machine
WSD	Word Sense Disambiguation

LIST OF FIGURES

Figure 2.1. Label Propagation Algorithm ……...4

Figure 3.1. Diffusion Kernel Algorithm ……10

Figure 4.1. Diffused Label Propagation (DILP) Algorithm ……..11

LIST OF TABLES

Table 5.1. For baseball-hockey class and tfidf ……..15

Table 5.2. For pc-mac class and tfidf ……15

Table 5.3. For religion-atheism class and tfidf ..……16

Table 5.4. For baseball-hockey class and tf ……...17

Table 5.5. For pc-mac class and tf ……18

Table 5.6. For religion-atheism class and tf ……...18

Table 5.7. Interest data, lp vs. grf ……...19

Table 5.8. Hard data, lp vs. grf ……...20

Table 5.9. Serve data, lp vs. grf ……..20

Table 5.10. Line data, lp vs. grf ……..21

Table 5.11. WSD Datasets ……..22

Table 5.12. Interest Data with all sense ……..23

Table 5.13. Hard Data with all sense ……..25

Table 5.14. Line Data with all sense ……..27

Table 5.15. Serve Data with all sense ……29

Table 5.16. Interest Data with binary sense ……...31

Table 5.17. Line Data with binary sense ……33

Table 5.18. Serve Data with binary sense ……..35

Table 5.19. Hard Data with binary sense ……...37

Table 5.20. Classification Accuracy Results on Interest, Line, Hard and Serve Datasets 40

1. INTRODUCTION

There are many words that have many meanings in every language. For example, the noun *nail* represents a part on our fingers and toes which is called fingernail and toenail, more specifically. Moreover, *nails* represent thin, sharp metal pieces used in construction. Another example of a word with more than one meaning is *jam*. One possible sense of the word *jam* is a sweet spread made from fruit and sugar boiled. When used as a verb *jam* means to put something into a space that is too small for it. Another meaning of the word *jam* is the squeezing into a specified space, it is used for traffic and called as traffic jam.

Humans subconsciously understand the right meaning easily by observing the context. On the other hand to computationally identify the suitable sense of the word is not a trivial task. Word Sense Disambiguation (WSD) is the issue of determining which sense is the correct sense of a given word according to its context [1].

WSD is a very popular task in both academic and non-academic platforms. There is an extensive bibliography for resolving ambiguity of polysemous words. There are two forms of WSD tasks, lexical-sample task and all-words task [2]. The main difference between lexical-sample task and all-words task is that the latter attempts to disambiguate all types of words (i.e., adjectives, adverbs, nouns, verbs …etc.) in the entire corpus while lexical-sample task deals with only pre-selected types of target words.

In order to resolve ambiguity several algorithms have been developed. It Makes Sense (IMS) for WSD of English text is a supervised system. Zhi and Ng [3] attempt to handle the WSD task with the IMS framework. In this framework the support vector machine (SVM) classifier is used and multiple features such as part-of-speech (POS) are integrated, surrounding words and local collocations. Taghipour and Ng [4] aim to incorporate word embeddings in a continuous space to IMS framework in a WSD system [5]. Iacobacci et al. [6], also use IMS framework and word embeddings as features and report that significant performance improvements are achieved. Melamud et al. [7] use context2vec with bidirectional long short-term memory (Bi-LSTM) to train word vectors. According to the experimental results, they get better results on the WSD task. Other efforts including LSTM and bi-LSTM are [8, 9, 10]. For instance the methodology in [8] has three layers; the softmax layer, a hidden layer and a Bi-LSTM layer. Bi-LSTM and hidden layer share parameters over all word types and senses. The outputs of left pass vector and right pass vector of LSTM are concatenated and this concatenation forms the output of the model. There is another model called gloss augmented

WSD neural network (GAS) is introduced by [11].

We consider the WSD as a transductive classification problem and propose an improvement to Label Propagation which is a well-known transductive inference algorithm. Our approach make use of a semantic diffusion kernel. We name this new algorithm as diffused label propagation algorithm (DILP). We evaluate our proposed algorithm with experiments utilizing various sizes of training sets of disambiguated corpora accompanied with unlabeled data. These experiments help to answer the following questions: 1. Does our algorithm with semantic kernel formulation yield higher classification performance than the popular kernels? 2. Under which conditions does a kernel design perform better than others? 3. What kind of regularization methods result in better performance? Our experiments demonstrate that our approach can outperform baseline methods which are RBF kernel, Linear kernel, and Polynomial kernel in terms of accuracy under we describe below.

2. BACKGROUND

2.1. Word Sense Disambiguation

Word Sense Disambiguation (WSD) actually is a classification problem since it attempts to find the most suitable sense of a word among its possible senses defined by a dictionary based on its context. There are two variants of WSD tasks [2]. One is the lexical sample task. In the Lexical sample task, there are only a small number of pre-selected types of target words. There is an inventory of senses of each word. Supervised machine learning approaches are used to train a classifier for each word. The second type of WSD task is the all-words task. The all-words-task tries to disambiguate all types of words (i.e., nouns, adjectives, verbs, adverbs…etc.) in the entire corpus. This all-words-task generally needs wide-coverage systems [2].

2.2. Label Propagation Algorithm

Label Propagation (LP) [12] is a semi-supervised and transductive learning method. It is used to label of all nodes in a graph structure. That means learning is limited to the given data. There are inductive methods which aims to model the whole sample space [13, 14].

There are two main assumptions behind the LP algorithm. First, the closer nodes tend to have similar class labels which is similar to the K-Nearest Neighbor (KNN). Second, the points on the same cluster have the same label [15]. The first one is a local whereas the second one is a global assumption. These two assumptions together are called the cluster assumption [16, 17].

The algorithm propagates the class label of a node in a graph to neighboring nodes by using weighted edges and proximity. Until convergence to a unique solution is achieved, this process is repeated, which is guaranteed [12]. In order to assign labels to the unlabeled data, the labeled data is used in a classification model.

In this study, we employed the well-known LP algorithm [12], which is implemented in the Scikit-Learn Library[1]. We explain the problem setup and the steps of the algorithm.

For $l, u \in \mathbb{N}^+$ with $l < u$ and $l + u = n$ we let $X = \{x_1 \dots x_n\}$ denote the array of contexts (documents) and each x_i is a vector of dimension t where t denotes the number of terms. Hence X is a $term \times document$ matrix. We let class labels to denote senses of words to be disambiguated where $Y_L = \{y_1 \dots y_l\}$ denotes the known senses (class labels), $Y_U =$

[1] https://scikit-learn.org/stable/#

$\{y_{l+1} \ldots y_{l+u}\}$ denotes the unknown senses and $Y = Y_L \cup Y_U$.

First, we form a fully connected graph whose vertices are $(x_1, y_1) \ldots (x_{l+u}, y_{l+u})$. The edges between the vertices, say i, j are weighted by using similarity score between the vectors x_i and $x_{j,}$ which is denoted by w_{ij}. The simplest similarity score is the dot product of the vectors. Another popular way of computing the similarity is the radial basis function.

$$w_{ij} = e^{-\frac{||x_i - x_j||^2}{2\sigma^2}} \quad (1)$$

It results in larger weights when the distance between two vectors is closer. Radial Basis Function (RBF) is commonly used among other graph-based methods [12, 15, 18, 19].

The sigma parameter controls the weights. During the testing phase of this work we observed that the change in σ effects the model substantially, this is a well-observed phenomenon [17].

There are two steps for LP model learning. In the first step, we use the similarity scores of the vectors in order to assign weights between nodes. Then we form the diagonal degree matrix, D_{ii} as follows.

$$D_{ii} = \sum_j W_{ij} \quad (2)$$

Bengio et al. claim that the algorithm works better if the diagonal elements of W are set to zero [20]. In the second part, the labels (senses in our case) are propagated until convergence. The initialization of Y_U is not important but for Y_L, we used two senses as labels. We choose the initialization of Y as $Y_0 = \{y_1 \ldots y_l, 0, \ldots, 0\}$. After the initialization, the algorithm is as follows.

Iterate

$\hat{Y}^{t+1} \leftarrow D^{-1} W \hat{Y}^t$

$\hat{Y}_L^{t+1} \leftarrow Y_L$

Until convergence

Fig 2.1. Label Propagation Algorithm

Algorithm 1 returns the propagated labels which are then used to label the points in X. The matrix Y can be constructed differently when there are more than two classes as in [21]. For WSD, Niu et. al. 2005 used a $document \times class$ matrix with $Y_{ij} = 1$ if y_i is in sense tag s_j and $Y_{ij} = 0$ otherwise [21]. Then they use the same algorithm in a one-versus-rest fashion.

Some other graph-based learning methods that are well-known are as follows: [19] is based on Markov random walks, [22] iteratively computes the distributions of labels (MAD algorithm), [23] is similar to Markov random walks, [17] computes local structure and combines them to get a better global learner.

They generally have similar graph constructions but different regularizing methods or loss function. For an extensive survey on the topic one can refer to [20] or [24]. There are exceptions that focus on graph connectivity, for example, [1] studies extensively the structure of a given graph for unsupervised learning, but they are beyond the scope of his work.

3. RELATED WORK

3.1. Approaches for WSD

There are several types of algorithms developed for WSD in the literature.

The IMS Framework: It Makes Sense (IMS) is a supervised English all-words WSD system. Zhi and Ng [3] present the IMS framework with the default support vector machine (SVM) classifier. Moreover, they integrate multiple features such as part-of-speech (POS), surrounding words and local collocations. It is a flexible framework which allows users to choose different features, classifiers and preprocessing tools.

IMS$_{+emb}$: Taghipour and Ng [4] aim to incorporate word embeddings which are representations of words in a continuous space [5] in a WSD system. In their work they choose to work with the word embedding that is created by Collobert and Weston [26] which they name as CW. To produce a word representation using CW embedding, they use a feed forward neural network with stochastic gradient descent as the training algorithm. The WSD framework in use is IMS and they also construct the default settings of IMS as baseline. The idea behind using word embeddings in the WSD task is to add the distribution of the representation of the words to the system which leads the classifier to take the similarity of the words into account.

One other IMS based WSD method is Iacobacci et al. [6], who show that by designing IMS properly and employing word embeddings as features gives significant performance improvements which are hard to beat [11].

context2vec: Melamud et al. [7] introduce a different word embedding which they call context2vec. Their basic idea is to use bidirectional LSTM (Bi-LSTM) to train a word vector. Their model is based on word2vec's CBOW architecture [5] but they use Bi-LSTM instead of averaged word embeddings (AWE). They outperform AWE and they get similar or better results on WSD task (and some other tasks such as sentence completion and lexical substitution).

LSTM and Bi-LSTM: Long Short-Term Memory (LSTM) is a kind of gated recurrent neural network [27] which is introduced for sequence models to make use of the long term dependencies, i.e. it preserves information from the preceding inputs (past). The bidirectional LSTM (Bi-LSTM) [28] is an adaptation of the LSTM which runs inputs both from the past and the future. As humans understand the sense of a word by observing the context, this WSD technique uses information about both preceding and succeeding words [8]. Their model has three layers; the softmax layer, a Bi-LSTM layer and a hidden layer. Bi-LSTM and hidden

layer share parameters over all word types and senses. The output of the model is the concatenations of the outputs of left pass vector and right pass vector of LSTM. Furthermore, there is an embedding layer that provides out the word embedding, which is GloVe [5] in their case. In order to decrease the dependency on individual words in the training set, they use a regularization technique [29]. Compared to $IMS_{+adapted}$ CW and other naive Bi-LSTM methods, their model achieves state-of-the-art level improvements. One of the most important advantages of this model is that it is independent of the chosen language.

The model gloss augmented WSD neural network (GAS) is introduced by Luo et al. [11]. The main differences between this model and the Bi-LSTM model of [8] are the gloss module and the memory module of GAS. The gloss module makes use of the glosses (sense definitions) which is ignored by previous neural network models. This module can be seen as an embedding module of glosses which is trained by a Bi-LSTM. The aim of the memory module is to model the relationship between the glosses and the context. Experiments show that their model outperforms Raganata et al. [9], Kägeback and Salomonsson [8] and Iacobacci et al. [6].

3.2. Label Propagation Algorithms

Generally, in graph based learning algorithms, the graph constructions are similar [24] as we mentioned before. If we consider the given graph method as estimating a classification function f, then the graph based algorithms differ by their loss function (how close f estimates labels to the given ones) or regularizers (how smooth is f on the whole graph).

For example, the methods with combinatorial graph Laplacian $\mathcal{L}$, are MinCuts [23], label propagation [12], and Gaussian random fields [12, 18]. The method that employs usage of kernels on graphs is manifold regularization [13]. Discrete regularization [15] uses normalized graph Laplacian. Interpolated and Tikhonov regularization employ L^p regularization [14].

In this study we also focus on regularization of our graph based method by using a Kernel and a regularization method based of graph Laplacian. Before explaining graph Laplacian, let us discuss why such regularization is needed. As mentioned in several studies [13, 14, 15, 24], the classifier learned by the graph-based method has to be consistent with both the initial labeling and the geometry of the data. Rapid changes between close points cause the model to fail to fulfill the cluster assumption and it has to be punished for that.

One common way is to normalize the graph Laplacian as given in [15]. We add a regularization step in the initialization.

$$\mathcal{L} \leftarrow D^{-1/2} W D^{-1/2} \tag{3}$$

Then we iterate the following equation in the label propagation algorithm where Y_0 is the initialization and t denotes the iteration steps [12].

$$Y^{t+1} \leftarrow \alpha \mathcal{L} Y^t + (1-\alpha) Y_0 \tag{4}$$

This new regularized algorithm is called label spreading [15, 20]. For a proof of the convergence of this algorithm, one can refer to [15]. Note that the eigenvalues of $\alpha\mathcal{L}$ are in $(-1,1)$.

Another method that we mention here is the Kernel method. Kernels among nodes are referred to as kernels on graphs and it should not be mixed with graph kernels [30]. The most popular kernel used for label propagation is the radial basis kernel or Gaussian kernel,

$$K(x, x') = \frac{1}{2\pi\sigma^2} \exp\left(\frac{-||x-x'||^2}{2\sigma^2}\right) \tag{5}$$

where the scaling parameter is σ. By some parameters change, it is the solution of the diffusion equation (Kondor et. al. 2004). In physics, the diffusion equation describes how heat and gas diffuse in homogenous environment with time. Intuitively, in graph based learning methods, we want the given labels to diffuse over the whole graph. This intuition explains why Gaussian kernels are very popular in graph based learning methods. For a formal proof, one can refer to [13, 15, 31]. The heat or gas equations are all in a continuous setting, but our problem is a discrete one. In [31, 32] a discretization of a Gaussian kernel by using the graph Laplacian is introduced that is called the diffusion kernel

$$K_\lambda = exp(\lambda\mathcal{L}) = \lim_{s\to\infty} (I + \frac{\lambda\mathcal{L}}{s})^s = I + \lambda\mathcal{L} + \frac{\lambda^2\mathcal{L}^2}{2} + \cdots \tag{6}$$

where $\mathcal{L}$ is the graph Laplacian of the graph and λ is control parameter. Note that, the powers of positive semi definite matrices are positive semi-definite and hence positive definiteness is

assured.

Semantic Diffusion Kernel: This idea of using the diffusion kernel in a discrete setting such as text classification and particularly in word sense disambiguation is not new. Very recently, Wang et al. [25] proposed a semantic diffusion kernel which uses semantic similarities as a diffusion process in a graph whose nodes represent the contexts and edges represent the first-order similarities.

Diffusion considers all possible paths connecting two nodes in the graph. The basic idea of the algorithm is to build an augmented term-document matrix. Then by applying the "Diffusion" process to this matrix so that the words which are in the same class become closer to each other.

In [25, 33] they used this technique with support vector machines to capture the higher order correlations between terms. Let us explain how they created their diffusion matrix in detail. First, they get a $term \times term$ matrix $G = X\,X^T$ and compute the following.

$$S = \exp\left(\frac{\lambda}{2}G\right) = \frac{1}{2}\left(2I + \lambda G + \frac{\lambda^2 G^2}{2!} + \cdots\right) \tag{7}$$

The matrix S is called a semantic similarity matrix. Then they compute $document \times document$ semantic diffusion kernel $K_\lambda = X^T S^T S X$, which they used for SVM. According to their experiments their diffusion kernel is superior to LSI and linear kernel with several SensEval disambiguation tasks such as interest, line, hard and serve. The Diffusion algorithm is shown in the following Figure.

Data: Matrix X, Parameter λ, Step
Result: Semantic Diffusion Kernel
initialization;
Calculate $G = XX^T$;
Compute the Taylor expansion as follows;
S : Identity matrix of the same size as G;
for $I = 1$ ***to*** *Step* **do**
  $S := S + \frac{\lambda^I}{I!}G^I$;
end
Compute $X^T S^T S X$;

Fig 3.1. Diffusion Kernel Algorithm

Other semi-supervised WSD studies with LP are [21] that uses label propagation with similarities (cosine, Jensen-Shannon divergence) and get very good results, [34], which uses basic dot product with LP, on parallel corpora and applies on Chinese data sets. Recent works generally focus more on merging label propagation method with deep learning methods such as LSTM [35]. As far as we know, none of them used semantic diffusion kernels in their work.

4. APPROACH

In our work, we apply label propagation for the WSD task. First we get dataset from SensEval and extracted four different dataset which are interest, line, hard and serve. For each datasets we have chosen two major senses. After that we have calculated similarity by using dot product. Then the Laplacian regularization method is employed to keep the local global consistency. Then we used the DILP instead of RBF kernel to capture the semantic similarities and to get a smooth decision surface. Using dot product, Laplacian regularization and DILP combination completed our study. Finally, we compared our results with the RBF kernel. According to the experimental results on WSD task, DILP gives higher classification accuracy in compare to RBF. Following figure shows the detail about DILP.

Data: Matrix X, Parameter λ, Step
Result: Propagated Labels
initialization;
Calculate the affinity matrix with dot product $G = XX^T$;
Calculate the diagonal degree matrix D, with $D_{ii} = \sum_j G_{ij}$;
Calculate the graph Laplacian $L = D^{-1/2}GD^{1/2}$;
Compute the Taylor expansion as follows;
W : Identity matrix of the same size as L;
for $I = 1$ ***to*** *Step* **do**
  $W := W + \frac{\lambda^I}{I!}L^I$;
end
Call Algorithm Label Propagation for W;

Fig 4.1. Diffused Label Propagation (DILP) Algorithm

5. EXPERIMENTS AND RESULTS

Experimental codes are written by using some frameworks for machine learning. We have started our experiments by using weka then given up it. Second and the last framework is the scikit learn.

5.1. Weka vs. Scikit Learn

Weka is machine learning software written with Java. It contains many tools for machine learning processes such as data preparation, classification, regression, clustering, association rules mining, and visualization.

Some experiments are made with Weka. However it has some disadvantages. One of the handicaps is lacking in machine learning API. Some APIs have to be written manually. Another disadvantage is being Java based platform. Java does not have enough scientific computing packages. So scientific parts are also have to be written manually. These deficiencies are time consuming therefore weka is being changed with scikit learn.

Python programming language is a scripting language. Learning velocity is so high and installation is so easy. It has many advantages for machine learning (ML) world. It has very powerful scientific computing library named numpy. Numpy is so fast and has many API's for scientific calculations. For machine learning there is also a framework for python which is scikit-learn. Scikit-learn is the most preferred platform in ML world. It has many capabilities for ML.

5.2. Gaussian Random Fields (GRF)

Our first aim was exceed the accuracy of label propagation(LP) algorithm with RBF kernel.

In scikit-learn, label propagation classifier has many parameters. These are kernel, gamma, number of neighbors, alpha, maximum iteration, and convergence tolerance. Also in scikit-learn these parameters are optimized. So exceeding LP algorithm with these optimized parameters is main aim in this thesis.

In order to exceed LP, we used GRF algorithm. First we calculate document similarity matrix. In order to keep label values of these documents we keep a vector which is named y_hat vector. In this vector unlabeled values are set as zero. Then get dot product of document similarity matrix with y_hat vector. Finally weight vector is calculated and dot product that gotten above divided by this weight vector. The result of this calculation is equal to y_hat.

5.2.1. GRF Parameters

There are some parameters that affect the result of the GRF algorithm. They are:

- Similarity Method
- Vectorizer
- Iteration Count
- Converge Limit

5.2.2. Similarity Methods

For GRF calculation we first calculate document similarity matrix by using train data.

Similarity methods affect the accuracy of this algorithm. So we tested our algorithm with these different similarity methods. These are:

I. Cosine Similarity

II. Dot Similarity

III. Jaccard Similarity

The best result obtained by dot similarity. Therefore our experiments are based on dot similarity and others are ignored.

5.2.3. Vectorizer

Feature extraction is the module in scikit learn framework. It has many feature extractors that named vectorizer. We have used Tf, tfidf, doc2vec, and fasttext. The best result is given by tfidf vectorizer.

5.2.4. Iteration Count

Different size of iteration count, from 10 to 10000 was tried. Hence approximately after 100 iteration the result was not changed.

5.2.5. Converge Limit

GRF formula is iterated until converge limit exceeded. Generally we used 0.001 for converge limit which means the change of y_hat vector. Eventually the final y_hat represents the predicted labels for unlabeled instances.

5.3.Label Propagation

Label Propagation algorithm is baseline algorithm for this thesis. It is a semi-supervised machine learning algorithm. It uses labeled instances to assign labels to the unlabeled ones.

Labels are propagated to the unlabeled instances throughout the algorithm [1]. In scikit learn it has optimized parameters. These are:

- Kernel : RBF
- gamma: 20
- neighbors : 7
- alpha : None
- max iteration : 1000
- tolerance : 1e - 3

Above parameters are optimized parameters and give the best result for this algorithm. So we have made our experiments by using these parameters too.

5.4. Label Propagation vs. GRF

First we have execute algorithms based on 20 news group data. In these experiments binary classes are used because in article these classes are used [12]. Result of this experiment is shown below:

Table 5.1. For baseball-hockey class and tfidf

labeled ratio	algorithm	accuracy(%)
5%	GRF	50.08
5%	LP	**86.96**
20%	GRF	77.14
20%	LP	**95.57**
30%	GRF	90.45
30%	LP	**97.15**
40%	GRF	92.90
40%	LP	**97.82**

For baseball and hockey class experiments we conclude that while training ratio increases, accuracies of both LP and GRF increase too. But LP exceeds GRF for all steps.

Table 5.2. For pc-mac class and tfidf

labeled ratio	algorithm	accuracy(%)
5%	GRF	55.31
5%	LP	**79.02**
20%	GRF	74.55
20%	LP	**87.84**
30%	GRF	79.22
30%	LP	**90.32**
40%	GRF	83.16
40%	LP	**92.80**

This experiment uses pc and mac classes. Accuracies increase for each step and for each algorithm. However GRF can not exceed LP for any step.

Table 5.3. For religion-atheism class and tfidf

labeled ratio	algorithm	accuracy(%)
5%	GRF	55.65
5%	LP	**71.52**
20%	GRF	71.41
20%	LP	**88.44**
30%	GRF	81.21
30%	LP	**92.41**
40%	GRF	59.61
40%	LP	**80.04**

The last experiment represents the religion and atheism class values. GRF again can not exceed the LP. There is a strange result on 40% step. The accuracies of both algorithms decreased after 30%. But decrease ratio of GRF algorithm is so high. Hence LP exceeds it.

As seen above tables, the results show that GRF can not exceed the LP. Because of that we have made new experiments. Below experiments are based on tf vector. After failing on tfidf vector experiments, we decided to use tf instead of tfidf. Below tables show the results for each class pairs.

Table 5.4. For baseball-hockey class and tf

labeled ratio	algorithm	accuracy(%)
5%	GRF	50.08
5%	LP	**52.80**
20%	GRF	62.00
20%	LP	**86.80**
30%	GRF	85.68
30%	LP	**94.90**
40%	GRF	86.36
40%	LP	**95.90**

Compare to the tfidf, the results of tf are more low. Moreover LP gives higher accuracy for each experiment steps.

Table 5.5. For pc-mac class and tf

labeled ratio	algorithm	accuracy(%)
5%	GRF	**78.38**
5%	LP	71.74
20%	GRF	62.56
20%	LP	**78.93**
30%	GRF	72.25
30%	LP	**85.70**
40%	GRF	77.46
40%	LP	**89.46**

GRF algorithm is more successful than LP for only 5% training ratio. After that step for 20% trainig ratio, its accuracy decreases. Then it shows increasing scenario. However, excepy first experiment step LP exceeds the GRF.

Table 5.6. For religion-atheism class and tf

labeled ratio	algorithm	accuracy(%)
5%	GRF	55.55
5%	LP	**58.10**
20%	GRF	60.07
20%	LP	**64.76**
30%	GRF	68.65
30%	LP	**69.42**
40%	GRF	56.31
40%	LP	**74.44**

This religion&atheism class values can not affect the behavior of the GRF algorithm. Again we see that GRF can not exceed the LP algorithm.

5.5. GRF vs. Label Propagation with WSD Data

After 20news group experiments, data is changed with wsd data. WSD data has four different senses (interest, hard, serve, and line). Senses are the meanings of the words. For instance the word intereset has six different senses(meanings) in interest dataset. Experiments are made based on these all senses. The results can be seen below:

Table 5.7. Interest data, lp vs. grf

labeled ratio	algorithm	accuracy(%)
5%	GRF	50.54
5%	LP	**56.25**
10%	GRF	50.49
10%	LP	**55.50**
20%	GRF	50.38
20%	LP	**56.00**
30%	GRF	50.11
30%	LP	**55.75**
40%	GRF	50.43
40%	LP	**56.50**

For interest data, LP always gives higher accuracy than GRF. While training ratio increases, the expectation was increasing accuracy but that not occured. For each experiment step, accuracies change.

Table 5.8. hard data, lp vs. grf

labeled ratio	algorithm	accuracy(%)
5%	GRF	79.76
5%	LP	**79.79**
10%	GRF	**79.74**
10%	LP	79.67
20%	GRF	**79.75**
20%	LP	79.67
30%	GRF	**79.85**
30%	LP	79.79
40%	GRF	79.66
40%	LP	**79.79**

Table 5.9. serve data, lp vs. grf

labeled ratio	algorithm	accuracy(%)
5%	GRF	41.53
5%	LP	**55.37**
10%	GRF	41.48
10%	LP	**55.60**
20%	GRF	41.49
20%	LP	**58.00**
30%	GRF	41.53
30%	LP	**59.61**
40%	GRF	41.44
40%	LP	**62.24**

Table 5.10. line data, lp vs. grf

labeled ratio	algorithm	accuracy(%)
5%	GRF	52.33
5%	LP	**57.03**
10%	GRF	52.32
10%	LP	**57.96**
20%	GRF	52.39
20%	LP	**58.75**
30%	GRF	52.34
30%	LP	**60.78**
40%	GRF	52.34
40%	LP	**62.50**

Above tables show that while labeled ratio increases the accuracy is not always increase too. Another important point is generally label propagation exceeds the grf but rarely grf can exceed it too. So there can not gotten any decision about performance of grf. As a result GRF algorithm is given up and tried new different algorithms.

5.6. DILP Algorithm Experiments and Results

DILP algorithm is described above and with wsd data new experiments are handled. In this experiment different types of kernels are used for Label Propagation algorithm. As described above, the baseline is RBF kernel which is optimized and gives the best result. Hence, that kernel changed and different kernels are used which are:

- Taylor Kernel
- FNormalization Kernel
- VNormalization Kernel
- Taylor + FNormalization Kernel
- Taylor + VNormalization Kernel

5.6.1. WSD Datasets

For the purpose of measuring our algorithm we select the datasets namely hard, line, serve and interest from SensEval. This dataset is used in the studies of WSD. Word occurences in all dataset were manually tagged with a WordNet sense.

Hard Data: There are 4333 instances in hard data. These instances are labeled as HARD1, HARD2 and HARD3.

Line Data: The line data contains 4146 instances. Six different senses and frequencies are shown on Table 11.

Serve Data: 4378 instances occur in this dataset. Moreover Table 5.11 shows four different senses and their frequencies.

Interest Data: It has 2367 instances. There are six senses in this dataset and their details are shown on Table 11.

Table 5.11. WSD Datasets

Name	***No***	***Sense***	***Frequency***
Hard	1	not easy (difficult)	3455
	2	not soft (metaphoric)	502
	3	not soft (physical)	376
Line	1	Stand in line	349
	2	A nylon line	373
	3	A line between good and evil	374
	4	A line from Shakespeare	404
	5	The line went dead	429
	6	A new line of workstations	2217
Serve	1	Function as something	853
	2	Provide a service	439

Table 5.11. WSD Datasets (Continue)

	3	Supply with food	1814
	4	Hold an office	1272
Interest	1	Readiness to give attention	361
	2	Quality of causing attention to be given	11
	3	Activity, etc. That one gives attention to	66
	4	Advantage, advancement or favor	177
	5	A share in a company or business	500
	6	Money paid for the use of money	1252

First experiments have been performed for all sense data of datasets which are hard, serve, line and hard data.

Table 5.12. Interest Data with all sense

Ratio Labeled	**Algorithm**	**Accuracy (%)**
10	RBF	**67.25**
10	Taylor	53.65
10	FNormalization	52.89
10	VNormalization	53.35
10	FNormalization + Taylor	53.65
10	VNormalization + Taylor	57.54
20	RBF	**73.72**
20	Taylor	55.47
20	FNormalization	53.02
20	VNormalization	53.73
20	FNormalization + Taylor	55.47

Table 5.12. Interest Data with all sense (Continue)

20	VNormalization + Taylor	62.27
30	RBF	**77.22**
30	Taylor	58.34
30	FNormalization	53.82
30	VNormalization	56.18
30	FNormalization + Taylor	58.34
30	VNormalization + Taylor	67.97
40	RBF	52.42
40	Taylor	63.49
40	FNormalization	57.03
40	VNormalization	60.28
40	FNormalization + Taylor	63.54
40	VNormalization + Taylor	**74.52**
50	RBF	60.70
50	Taylor	67.97
50	FNormalization	59.31
50	VNormalization	63.54
50	FNormalization + Taylor	68.01
50	VNormalization + Taylor	**79.63**
60	RBF	68.65
60	Taylor	72.20
60	FNormalization	62.06
60	VNormalization	67.21
60	FNormalization + Taylor	72.24

Table 5.12. Interest Data with all sense (Continue)

60	VNormalization + Taylor	**84.57**

The experiments on above tables are based on interest data with all sense. We have six different algorithms that they behave as kernel of LP algorithm. Moreover we have started with 10% training ratio up to 60%. The accuracies show that until 40%, RBF gives highest accuracy. But starting from 40% until 60%, Vnormalization + Taylor expansion exceeds the RBF.

Table 5.13. Hard Data with all sense

Ratio Labeled	**Algorithm**	**Accuracy (%)**
10	RBF	**81.74**
10	Taylor	80.17
10	FNormalization	79.75
10	VNormalization	79.85
10	FNormalization + Taylor	80.17
10	VNormalization + Taylor	**81.74**
20	RBF	**83.96**
20	Taylor	80.70
20	FNormalization	79.82
20	VNormalization	80.08
20	FNormalization + Taylor	80.70
20	VNormalization + Taylor	83.77
30	RBF	**85.92**
30	Taylor	81.19
30	FNormalization	80.06
30	VNormalization	80.42
30	FNormalization + Taylor	81.19

Table 5.13. Hard Data with all sense (Continue)

30	VNormalization + Taylor	85.87
40	RBF	87.92
40	Taylor	81.97
40	FNormalization	80.24
40	VNormalization	80.84
40	FNormalization + Taylor	81.95
40	VNormalization + Taylor	**88.02**
50	RBF	89.93
50	Taylor	82.80
50	FNormalization	80.42
50	VNormalization	81.21
50	FNormalization + Taylor	82.78
50	VNormalization + Taylor	**90.19**
60	RBF	92.01
60	Taylor	83.96
60	FNormalization	80.91
60	VNormalization	81.74
60	FNormalization + Taylor	83.89
60	VNormalization + Taylor	**92.24**

We can make some inferences with the interest data. RBF is successfull again until 40% training ratio. After 40%, Vnormalization + Taylor expanson become the best algorithm.

Table 5.14. Line Data with all sense

Ratio Labeled	Algorithm	Accuracy (%)
10	RBF	**60.97**
10	Taylor	53.61
10	FNormalization	53.47
10	VNormalization	53.59
10	FNormalization + Taylor	53.61
10	VNormalization + Taylor	58.05
20	RBF	**65.84**
20	Taylor	54.05
20	FNormalization	53.54
20	VNormalization	53.83
20	FNormalization + Taylor	54.05
20	VNormalization + Taylor	62.71
30	RBF	**70.88**
30	Taylor	54.79
30	FNormalization	53.69
30	VNormalization	54.36
30	FNormalization + Taylor	54.79
30	VNormalization + Taylor	67.60
40	RBF	**74.74**
40	Taylor	56.22
40	FNormalization	54.10
40	VNormalization	55.40
40	FNormalization + Taylor	56.22

Table 5.14. Line Data with all sense (Continue)

40	VNormalization + Taylor	72.35
50	RBF	**78.94**
50	Taylor	57.71
50	FNormalization	54.53
50	VNormalization	56.77
50	FNormalization + Taylor	57.71
50	VNormalization + Taylor	77.35
60	RBF	**82.94**
60	Taylor	59.96
60	FNormalization	55.35
60	VNormalization	58.53
60	FNormalization + Taylor	59.96
60	VNormalization + Taylor	82.36

Experiments of line data shows unsuccessful results for our algorithms. Because for all training ratio step, RBF gives highest accuracy.

Table 5.15. Serve Data with all sense

Ratio Labeled	Algorithm	Accuracy (%)
10	RBF	**67.38**
10	Taylor	42.71
10	FNormalization	41.43
10	VNormalization	41.84
10	FNormalization + Taylor	42.71
10	VNormalization + Taylor	47.28
20	RBF	**72.65**
20	Taylor	53.06
20	FNormalization	50.13
20	VNormalization	57.81
20	FNormalization + Taylor	53.06
20	VNormalization + Taylor	65.09
30	RBF	**76.29**
30	Taylor	62.56
30	FNormalization	60.02
30	VNormalization	63.40
30	FNormalization + Taylor	62.56
30	VNormalization + Taylor	73.38
40	RBF	**80.42**
40	Taylor	67.19
40	FNormalization	63.75
40	VNormalization	65.50
40	FNormalization + Taylor	67.19

Table 5.15. Serve Data with all sense (Continue)

40	VNormalization + Taylor	78.39
50	RBF	**84.12**
50	Taylor	70.07
50	FNormalization	64.86
50	VNormalization	67.03
50	FNormalization + Taylor	70.07
50	VNormalization + Taylor	82.66
60	RBF	**87.89**
60	Taylor	72.75
60	FNormalization	65.98
60	VNormalization	68.50
60	FNormalization + Taylor	72.75
60	VNormalization + Taylor	86.29

For all sense experiments, DILP algorithm is better than RBF on interest and hard data with 40%, 50% and 60% ratio. However this is not valid for serve and line data. For these datasets for all ratios, RBF gives the best result. Hence experiments have to be continued. Next experiments are performed with binary classification not all classes.

Below tables show the binary classification results:

Table 5.16. Interest Data with binary sense

Ratio Labeled	Algorithm	Accuracy (%)
10	RBF	**72.70**
10	Taylor	61.90
10	FNormalization	59.34
10	VNormalization	60.74
10	FNormalization + Taylor	61.90
10	VNormalization + Taylor	62.71
20	RBF	**78.74**
20	Taylor	66.55
20	FNormalization	62.25
20	VNormalization	67.94
20	FNormalization + Taylor	66.55
20	VNormalization + Taylor	69.45
30	RBF	**81.41**
30	Taylor	72.59
30	FNormalization	67.82
30	VNormalization	75.60
30	FNormalization + Taylor	72.59
30	VNormalization + Taylor	77.58
40	RBF	67.59
40	Taylor	78.74
40	FNormalization	74.09
40	VNormalization	81.41
40	FNormalization + Taylor	78.74

Table 5.16. Interest Data with binary sense (Continue)

40	VNormalization + Taylor	**83.50**
50	RBF	73.86
50	Taylor	83.85
50	FNormalization	78.51
50	VNormalization	85.71
50	FNormalization + Taylor	83.85
50	VNormalization + Taylor	**88.03**
60	RBF	79.32
60	Taylor	87.57
60	FNormalization	81.30
60	VNormalization	88.50
60	FNormalization + Taylor	87.57
60	VNormalization + Taylor	**91.40**

After getting different results for all sense data, we have changed our experiments from all sense to the binary sense classifications. Interest data displays that for 40%, 50%, and 60% training ratios Vnormalization + Taylor Expansion algorithm gives the highest accuracy.

Table 5.17. Line Data with binary sense

Ratio Labeled	Algorithm	Accuracy (%)
10	RBF	**77.07**
10	Taylor	56.30
10	FNormalization	53.78
10	VNormalization	57.26
10	FNormalization + Taylor	56.30
10	VNormalization + Taylor	57.62
20	RBF	**81.03**
20	Taylor	61.10
20	FNormalization	57.74
20	VNormalization	68.78
20	FNormalization + Taylor	61.10
20	VNormalization + Taylor	69.50
30	RBF	**83.91**
30	Taylor	67.10
30	FNormalization	62.42
30	VNormalization	82.23
30	FNormalization + Taylor	67.10
30	VNormalization + Taylor	83.07
40	RBF	83.43
40	Taylor	74.06
40	FNormalization	69.62
40	VNormalization	87.39
40	FNormalization + Taylor	74.06

Table 5.17. Line Data with binary sense (Continue)

40	VNormalization + Taylor	**88.23**
50	RBF	86.31
50	Taylor	82.23
50	FNormalization	78.39
50	VNormalization	90.15
50	FNormalization + Taylor	82.23
50	VNormalization + Taylor	**90.87**
60	RBF	87.99
60	Taylor	88.35
60	FNormalization	84.63
60	VNormalization	91.47
60	FNormalization + Taylor	88.35
60	VNormalization + Taylor	**92.19**

Vnormalization + Taylor Expansion can not exceed the RBF until 40% training ratio. RBF is the best algorithm again.

Table 5.18. Serve Data with binary sense

Ratio labeled	Algorithm	Accuracy (%)
10	RBF	**87.00**
10	Taylor	60.40
10	FNormalization	58.78
10	VNormalization	60.30
10	FNormalization + Taylor	60.40
10	VNormalization + Taylor	63.15
20	RBF	**90.50**
20	Taylor	75.76
20	FNormalization	72.45
20	VNormalization	86.13
20	FNormalization + Taylor	75.76
20	VNormalization + Taylor	87.75
30	RBF	91.86
30	Taylor	87.71
30	FNormalization	85.22
30	VNormalization	91.18
30	FNormalization + Taylor	87.71
30	VNormalization + Taylor	**92.67**
40	RBF	93.77
40	Taylor	92.09
40	FNormalization	89.72
40	VNormalization	93.32
40	FNormalization + Taylor	92.09

Table 5.18. Serve Data with binary sense (Continue)

40	VNormalization + Taylor	**94.91**
50	RBF	94.97
50	Taylor	93.97
50	FNormalization	91.38
50	VNormalization	94.55
50	FNormalization + Taylor	93.97
50	VNormalization + Taylor	**96.43**
60	RBF	95.91
60	Taylor	95.43
60	FNormalization	92.90
60	VNormalization	95.10
60	FNormalization + Taylor	95.43
60	VNormalization + Taylor	**97.18**

The experiments of serve data shows different and better results for our algorithms. Until this data set, we have seen that our algorithms exceeds the RBF for only 40%, 50%, and 60%. But on this experiments our algorithm exceeds RBF on 30% training ratio too.

Table 5.19. Hard Data with binary sense

Ratio Labeled	Algorithm	Accuracy (%)
10	RBF	**74.82**
10	Taylor	63.09
10	FNormalization	61.84
10	VNormalization	66.28
10	FNormalization + Taylor	63.09
10	VNormalization + Taylor	66.74
20	RBF	**79.72**
20	Taylor	70.84
20	FNormalization	69.02
20	VNormalization	75.28
20	FNormalization + Taylor	70.84
20	VNormalization + Taylor	75.85
30	RBF	**84.28**
30	Taylor	78.92
30	FNormalization	76.87
30	VNormalization	83.59
30	FNormalization + Taylor	78.92
30	VNormalization + Taylor	84.05
40	RBF	87.69
40	Taylor	85.30
40	FNormalization	82.91
40	VNormalization	88.26
40	FNormalization + Taylor	85.30

Table 5.19. Hard Data with binary sense (Continue)

40	VNormalization + Taylor	**88.83**
50	RBF	79.27
50	Taylor	90.77
50	FNormalization	88.26
50	VNormalization	91.79
50	FNormalization + Taylor	90.77
50	VNormalization + Taylor	**92.59**
60	RBF	83.59
60	Taylor	94.19
60	FNormalization	91.00
60	VNormalization	93.62
60	FNormalization + Taylor	94.19
60	VNormalization + Taylor	**94.87**

We come to the same conclusion on this experimenst with interest and line data. Finally above binary classification experiments show that vnormalization + taylor kernel which is named as DILP give higher accuracy for 40%, 50%, and 60% labeled ratio. Finally we have tested our main experiment. In this experiment we compared RBF, DILP, Polynomial and Linear Kernel.

5.7. Experimental Setup

In order to analyze the accuracy performance of our approach we have used different sizes for training sets. Moreover we vary the labeled data percentage and report 40%, 50% and 60% where the performance difference of DILP is most visible. During experiments we optimized the parameters of DILP. For instance as similarity metric we use dot product and cosine similarity, and we choose dot product since it performs better.

For DILP there are three main parts, calculation the affinity matrix for dot product, normalization the graph Laplacian and Taylor expansion. We make some combinations for these three calculations and get crucial results. For example we use dot product with

normalization, dot product with Taylor expansion, only Taylor expansion, only normalization etc.

Our approach works best in the binary classification scenario. Therefore, for each dataset we select two of the largest senses (classes) whose sizes are similar. For interest dataset we chose senses 1 and 5, that are the second and third largest classes. For line dataset we choose senses 4 and 5, again second and third largest senses with similar number of instances. Using a similar approach, we select senses 2 and 3 for Hard dataset, and senses 2 and 6 for Serve dataset for our binary classification experiments.

5.8. Evaluation Results and Discussion

Table 5.20. represents the accuracy of each kernel for different training ratio of datasets. For each dataset we run experiments for 40, 50, and 60 training ratio. Our aim is presenting the higher accuracy of DILP opposed to other algorithms especially RBF. RBF is the baseline algorithm because for label propagation it gives the highest accuracy.

Especially this accuracy can be seen in scikit learn framework with optimized parameters. For interest data we can infer that 40% training data DILP gives the highest results. For 50% training data the result of RBF is 73.8% and DILP's result is 88.0%. But on the below row which has 50% training data, the result of DILP rises to 91.4% and the closest one is 86.1% .Other datasets, line, hard and serve also give the similar results. On serve data the results of RBF and DILP are very closed but the highest score still remains the same algorithm which is DILP. However, performance gain between baseline algorithm and DILP is calculated as:

$$Gain_{DILP} = (P_{DILP} - P_x) / P_x \quad (7)$$

where P_{DILP} is the classification accuracy of DILP and P_x represents the classification accuracy of RBF. Experimental results are also shown in Table II. The last column named gain in Table II, demonstrates the (%) of DILP over RBF computed as in Eq. (7).

As a result, DILP became more successful than RBF (baseline kernel) with high labeled ratio such as on below table. On the other hand linear and polynomial kernels can exceed the RBF but not the DILP. Table 5.20 shows the results of RBF, linear, polynomial and DILP kernels on serve data with different size of training data. Furthermore, it shows that DILP gives the best result. In the below table L. represents linear kernel, and Poly. shows the polynomial kernel. The last column Gain is the percentage gain of DILP over RBF. Moreover the stars near

gain values show the positive gain of DILP over RBF. The "*" represents the better performance of the DILP over RBF kernel based on Students t-Tests in the training sets.

Table 5.20. Classification Accuracy Results on Interest, Line, Hard and Serve Datasets

Dataset	***Labeled Data %***	***RBF***	***L.***	***Poly.***	***DILP***	**Gain**
Interest 1&5	40	67.5	73.9	76.0	**83.5**	23.7*
	50	73.8	78.3	81.4	**88.0**	19.2*
	60	79.3	81.1	86.1	**91.4**	15.2*
Line 4&5	40	83.4	69.6	70.9	**88.2**	5.7*
	50	86.3	78.3	78.9	**90.8**	5.2*
	60	87.9	84.6	87.5	**92.1**	4.8*
Hard 2&3	40	87.6	83.0	78.3	**88.8**	1.4*
	50	79.2	88.2	84.5	**92.5**	16.8*
	60	83.5	91.0	89.1	**94.8**	13.5*
Serve 2&6	40	93.7	89.7	80.1	**94.9**	1.3*
	50	94.9	91.3	86.1	**96.4**	1.6*
	60	95.9	92.9	91.2	**97.1**	1.2*

6. CONCLUSIONS AND FUTURE DIRECTIONS

The experiments show that our approach, the DILP algorithm performs considerably better than the baselines RBF, linear and polynomial in labeled data percentages of %40, %50 and %60 in binary classification scenario. One of the conclusions driven from this study is that dot similarity affects the score better than the cosine similarity. Another affect is about the Taylor step. Detecting the most efficient step has a crucial role on getting high score. The gamma value optimization also helps the score getting higher. One of the most effective optimization is using normalization with Taylor expansion. So these above optimization parameters make DILP, gives the best result in compare to RBF, linear and polynomial kernel.

As a future work, we extend DILP algorithm to perform multi-class classification using an appropriate method such as one versus all or one versus one approaches. Additionally, we aim to perform more experiments on other larger scale WSD corpora. We will also analyze how the other regularization methodologies with DILP improve classification accuracy in WSD in some test cases. An additional item in our agenda is to expand our approaches by adding further semantic-information and investigate how the construction of the used graph can affect the performance.

REFERENCES

1. R. Navigli and M. Lapata , “An Experimental Study of Graph Connectivity for Unsupervised Word Sense Disambiguation”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(4), pp. 678–692. 2010.
2. D. Jurafsky & J. H. Martin,“Speech and language processing“ (Vol. 3), London: Pearso, 2014.
3. Z. Zhi and H. T. Ng, “It makes sense: A wide-coverage word sense disambiguation system for free text”, In ACL 2010, Proceedings of the Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden, System Demonstrations, pp. 78-83.
4. K. Taghipour and H. T. Ng, “Semi- Supervised Word Sense Disambiguation Using Word Embeddings in General and Specific Domains”, In Proc. of NAACL-HLT, pp. 314 – 323, 2015.
5. T. Mikolov, K. Chen, G. Corrado, and J. Dean. “Efficient estimation of word representations in vector space”, arXiv preprint arXiv:1301.3781, 2013.
6. I. Iacobacci, M.T. Pilehvar and R. Navigli, “Embeddings forWord Sense Disambiguation: An Evaluation Study”, In Proc. of ACL, pp. 897-907, 2016.
7. O. Melamud, J. Goldberger, and I. Dagan, “context2vec: Learning Generic Context Embedding with Bidirectional LSTM”, In Proc. of CoNLL, pp. 51 – 61, 2016.
8. M. Kägeback and H. Salomonsson, “Word Sense Disambiguation using a Bidirectional LSTM”, In Proceedings of CogALex, pp. 51 – 56, 2016.
9. A. Raganato, C. D. Bovi, and R. Navigli, “Neural sequence learning models for word sense disambiguation,” In Conference on Empirical Methods in Natural Language Processing, pp. 1156-1167, 2017.
10. M. Le, M. Postma, J. Urbani and P. Vossen. 2018. A Deep Dive into Word Sense Disambiguation with LSTM. Proceedings of the 27th International Conference on Computational Linguistics, pages 354 - 365.
11. F. Luo, T. Liu, Q. Xia, B. Chang and Z. Sui, “Incorporating Glosses into Neural Word Sense Disambiguation”, Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pp. 2473 – 2482, 2018.

12. X. Zhu and Z. Ghahramani, “Learning from labeled and unlabeled data with label propagation”, Technical report, CMU CALD tech report, 2002.

13. M. Belkin, P. Niyogi and V. Sindhwani, “Manifold regularization: A geometric framework for learning from labeled and unlabeled examples", JMLR, 7, pp. 2399-2434, 2006.

14. M. Belkin, I. Matveeva & P. Niyogi , “Regularization and Semi-supervised Learning on Large Graphs”, In Proceedings of the 17th Conference on Learning Theory, 2004.

15. D. Zhou, O. Bousquet, T. Lal, J. Weston and B. Schölkopf, “Learning with local and global consistency”, Advances in Neural Information Processing System 16, 2004.

16. O. Chapelle, J. Weston & B. Schölkopf , “Cluster Kernels for Semi-Supervised Learning”, In Advances in Neural Information Processing Systems 15, 2003.

17. F. Wang and C. Zhang, “Label Propagation through Linear Neighborhoods.” IEEE Transactions on Knowledge and Data Engineering, 20(1), pp. 55–67. 2008.

18. X. Zhu, Z. Ghahramani and J. Lafferty, “Semi-supervised learning using Gaussian fields and harmonic functions”, ICML, 2003.

19. M. Szummer and T. Jaakkola, “Partially Labeled Classification with Markov Random Walks”, Advances in Neural InformationProcessing Systems 14, T.G. Dietterich, S. Becker, and Z. Ghahra-mani, eds., pp. 945-952, 2002.

20. Y. Bengio , O. Delalleau and N. Roux, “Semi-Supervised Learning”, chapter:Label Propagation and Quadratic Criterion, 2007.

21. Z- Y. Niu, D-H. Ji, & C.L. Tan, “Word sense disambiguation using label propagation based semi-supervised learning”, Proceedings of the ACL 2005.

22. P. Talukdar and K. Crammer, “New regularized algorithms for transductive learning”, In Machine Learning and Knowledge Discovery in Databases, pp. 442–457. Springer, 2009.

23. A. Blum and S. Chawla , “Learning from Labeled and Unlabeled Data Using Graph Mincuts”, Proc. 18th Int’l Conf. Machine Learning (ICML ’01), pp. 19-26, 2001.

24. X. Zhu, “Semi-supervised learning literature survey”, 2005.

25. T. Wang, J. Rao, Q. Hua, “Supervised word sense disambiguation using semantic diffusion kernel”, Engineering Applications of Artificial Intelligence, 27. Elsevier, pp. 167–174, 2014.

26. R. Collobert and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning”, In Proceedings of the 25th International Conference on Machine Learning, pp. 160-167, 2008.

27. S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory”, Neural Comput., 9(8), pp.1735-1780, 1997.

28. A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional lstm and other neural network architectures”, Neural Networks, 18(5), pp. 602-610, 2005.

29. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting”, The Journal of Machine Learning Research, 15(1), pp. 1929 – 1958 , 2014.

30. S. Vishwanathan, N. Schraudolph, R. Kondor and K. M. Borgwardt, “Graph kernels.The Journal ofMachine Learning Research”, 99, pp. 1201–1242, 2010.

31. R. I. Kondor, J.P. Vert, “Diffusion kernels." In: Schölkopf B, Tsuda K, Vert J P, editors. Kernel methods in computational biology. Cambridge (Massachusetts): MIT Press. pp. 171–192. 2004.

32. R. I. Kondor and J. Lafferty, “Diffusion kernels on graphs and other discrete input spaces”, In Proceedings of the ICML 2002.

33. T. Wang, W. Li, F. Liu and J. Hua, “Sprinkled semantic diffusion kernel for word sense disambiguation”, Engineering Applications of Artificial Intelligence., vol. 64, no. May, pp. 43–51, 2017.

34. M. Yu, S. Wang, C. Zhu and T. Zhao, “Semi-supervised learning for word sense disambiguation using parallel corpora”, 2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2011

35. D. Yuan, R. Doherty, J. Richardson, C. Evans, and E. Altendorf, “Semi-supervised Word Sense Disambiguation with Neural Models”, In Proc. of COLING, pp. 1374-1385, 2016.

RESUME

GÖKHAN KOCAMAN

Marmara University, Goztepe Campus

Technology Faculty, Computer Engineering Dept.

PK:34722, Kadikoy, Istanbul, Turkiye

Phone (Cell)+90-537-7683148

e-mail: gokhankocaman@marun.edu.tr, gokhankcmn@gmail.com

EDUCATION

M.S., Computer Science and Engineering Marmara University, Faculty of Engineering, Istanbul, Turkey, Ongoing

Thesis: Semi-supervised Classification of Documents via Label Propagation in Graphs

Advisor: Doç. Dr. Murat Can GANİZ

B.S., Computer Science and Engineering Marmara University, Istanbul, Turkey, 2010

WORK EXPERIENCE

2016 - ... FullStack Web Developer&Architect, Freelance, Istanbul, Turkey.

2013-2016 Senior FullStack Cloud&Web Developer, CODVIO, Istanbul, Turkey.

2010 - 2013 Java, Pl-Sql, Web, Mobile Developer, Aktek Bilişim, Istanbul, Turkey.

2009 - 2010 Java Developer, Inomera ARGE, Istanbul, Turkey.

RESEARCH INTERESTS

- Text Classification and Clustering, Sentiment Analysis

FOREIGN LANGUAGES

- English