GloDETC : Global Document Embeddings for Multi-label Text Classification

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of

B.Tech. - M.Tech. (Dual Degree)

by
Nitish Gupta
Roll No. : 10327461

under the guidance of
Prof. Harish Karnick
Prof. Rajesh M. Hegde

Department of Electrical Engineering
Indian Institute of Technology Kanpur
April, 2015

CERTIFICATE

It is certified that the work contained in this thesis entitled "*Merging Word Senses*", by *Sumit Bhagwani(Roll No. Y8127515)*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

(Prof. Harish Karnick)
Department of Computer Science and Engineering,
Indian Institute of Technology Kanpur
Kanpur-208016

June, 2013

Abstract

Currently used general purpose dictionaries are often too fine-grained, with narrow sense divisions that are not relevant for many Natural Language applications. WordNet, which is a widely used sense inventory for Word Sense Disambiguation, has the same problem. With different applications requiring different levels of sense granularities, producing sense clustered inventories of arbitrary sense granularity has evolved as a crucial task.

We try to exploit the resources available like human-labelled sense clusterings and semi-automatically generated domain labels of synsets, to estimate the similarity between synsets. Using supervision, we learn a model which predicts the probability of any two senses of a word to be merged. To learn a more generic model, we propose a graph based approach, which allows us to use the information learnt from supervision as well. Using this complete similarity measure, we propose a simple method for clustering synsets. We show that the coarse-grained sense inventory obtained significantly boosts the disambiguation of nouns on standard test sets.

Dedicated to
My Family

Acknowledgement

I wouldn't like to express my sincere gratitude towards my thesis supervisor, Prof. Harish Karnick, for his constant support and encouragement. I am grateful for his patient guidance and advice in giving a proper direction to my efforts. I am also grateful to Prof. Rajesh M. Hegde for giving me the freedom to work on a topic of my liking.

I am fortunate for the bevy of my friends I am a part of - I would like to thank them immensely, especially Aditya, Abhinav, Shubhendu, Parul, Pranay, Arzoo, Saurabh, Prajyoti, Asheesh and Anjana for their pleasant company and their ability to infuse humor into any situation. I would particularly like to thank Shrutiranjan Satapathy for his constant support, invaluable discussions and needless coffee breaks.

Last, but not the least, I would like to thank my parents and brother for their love and encouragement. Without their support and patience this work would not have been possible.

Nitish Gupta

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

With the advent of the Web, the world has seen a deluge of digital text. Word Sense Disambiguation(WSD) is a notoriously difficult problem in understanding text. Ambiguity is very common in text but humans are so competent at figuring out the word sense from context that most of the time they do not notice the ambiguity in the meaning of the word. Accurate sense disambiguation would aid a number of Natural Language applications; however it is widely acknowledged by WSD researchers that current accuracy levels of WSD systems need to be improved before they can be practically used in applications [**?**].

There are two major problems faced by the researchers in this area. One major problem is the dearth of sufficient training data for supervision. With a handful of sense-tagged texts currently available, existing WSD systems do not have examples for all the senses of a word most of the times. The other major problem that automatic disambiguation systems face is the fine-grained nature of the sense-distinctions in the sense inventory, WordNet in particular.

WordNet [**?**] [**?**] is well known to the Natural Language Processing community as a valuable resource and is one of the most widely used lexical resources. WordNet being a fine grained sense inventory makes it hard even for humans to reliably and consistently distinguish among word senses. In this thesis we would be addressing the latter problem and produce graded word sense relationships which can be used to derive coarse sense clusters with required granularity.

1.1 Motivation

The motivation for the work is two-fold: the variable needs of sense granularities by applications and saturation in fine-grained WSD performance.

1.1.1 Different granularity requirements of different tasks

The applicability of the work stems from the fact that different tasks and applications require different granularities of sense distinctions. The subtleties of sense distinctions captured by WordNet are helpful for language leaners [**?**] and in machine translation of languages as diverse as Chinese and English [**?**].

On the other hand, for tasks like Document Categorization [**?**] and Query Expansion [**?**], it may be sufficient to know if a given word belongs to a coarsely defined class of WordNet senses. Using fine grained sense inventory of WordNet may be detrimental to the performance of the performance of these applications. Thus developing a framework which can generate inventories with different granularities is a crucial task.

1.1.2 Saturation in Fine-Grained WSD

Lets observe the inter-tagger agreement(ITA) estimates of the data preparation for the Senseval/SemEval tasks on WSD: All Words(AW) or Lexical Sample(LS) in table 1.1.

Workshop and Task	WordNet Version	Verb	Noun	Adjective	Overall
Senseval-2 AW[1] [**?**]	1.7	70-80	NA	NA	NA
Senseval-2 LS [**?**]	1.7	NA	86.3	83.4	NA
Senseval-3 AW [**?**]	1.7.1	67.8	74.9	78.5	72.5

[1]Only approximate information on ITA is available

Senseval-3 LS [?]	1.7.1	NA	NA	NA	67.3
Semeval 2007 AW [?]	2.1	72	86	NA	NA

Table 1.1: Inter-tagger agreement for Various Data Preparations

It is a generally agreed upon fact that ITA serves as an upper bound on the performance of WSD systems. [?] estimates that unrestricted fine-grained WSD performance has a 70% upper bound. From Senseval and SemEval worskshops we observe that state-of-the-art automatic disambiguation systems indeed are not able to beat this bound. Therefore it seems that the major bottleneck in effective sense disambiguation is the fine grained nature of WordNet.

The above points are substantiated by the ITA achieved in preparation of gold standard datasets and performance of the systems in SemEval-2007 Task on Coarse grained WSD [?]. The ITA values on train and test datasets were 86.44% and 93.80%. These figures, compared to those in the table 1.1, show that the performance of the WSD systems can be improved by changing the granularity of the adopted sense inventory.

Some of the best systems of SemEval-2007 Coarse Grained WSD task achieved performances in the early 80s in the all words task and in the high 80s for the lexical sample task as compared to previous Senseval evaluation exercises, where state-of-the-art systems achieved performance far below 70%. This encourages us to study in depth the ideas for good sense clustering algorithms and use them to improve the WSD systems so that they can be used in practical scenarios.

1.2 WordNet

WordNet is a Lexical Knowledge Base (LKB) for the English language. The project was begun by George Miller, and is currently being maintained at Princeton University. WordNet is divided into 4 broad hierarchies based on POS, one each for nouns,

verbs, adjectives and adverbs. WordNet is similar to a thesaurus in the sense that synonymous word senses are grouped together into a single concept called *synset*. Every synset is described by a brief definition.

Synsets in WordNet are connected by relations, which can be categorized into two kinds:

- **Lexical Relations:** These are connections between word senses contained in respective synsets.
  - Antonymy: Synset A is an antonym of synset B if A and B have senses of opposite meaning.
  - Pertainymy: Adjective A is related to noun B by this relation if A "pertains to" B.
  - Nominalization: Noun A is related to verb B by this relation if A "nominalizes" B.
- **Semantic Relations:** These are connections between synsets as a whole.
  - Hypernymy: Synset A is a hypernym of synset B if B is a "kind of" A.
  - Hyponymy and Troponymy: Hyponymy is the inverse relation of hypernymy for nouns, while troponymy is the inverse relation of hypernymy for verbs.
  - Meronymy: Synset A is a meronym of synset B if A is a "part of" B.
  - Holonymy: The inverse relation of meronymy.
  - Entailment: Verb A entails verb B if A follows B.
  - Similarity: Relates similar adjectives.
  - Atribute: Noun A is related to adjective B by this relation if A serves as an attribute of B.
  - See also: Applicable if two adjectives are "related" semantically.

Figure 1.1 shows a subgraph of WordNet [**?**]. WordNet 3.0 contains 155,287 words organized into a total of 117,659 synsets. [2]

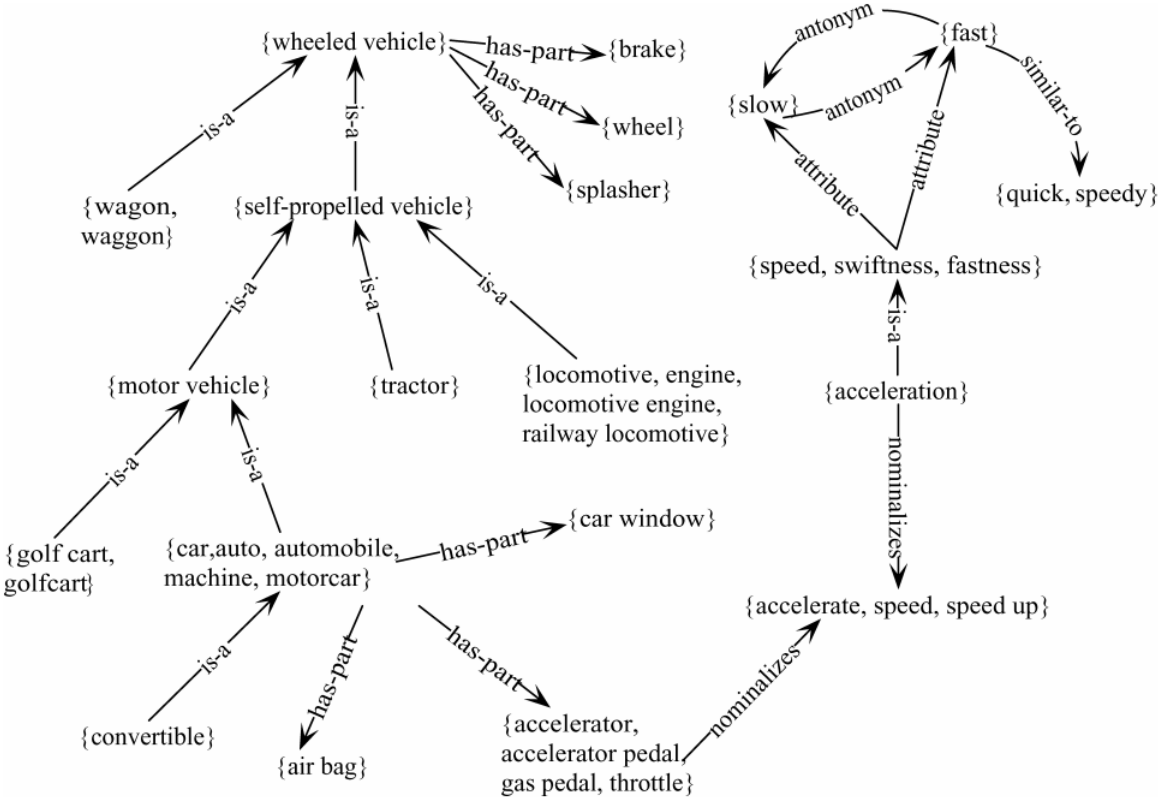


Figure 1.1: Excerpt From WordNet

1.3 Problem Statement

Formally, the task we are attempting has two objectives:

1. Given a fine-grained sense inventory like WordNet, we wish to produce a clustering over WordNet synsets at any arbitrary granularity, which can serve as a coarse sense dictionary.

2. Given a coarse sense inventory obtained by clustering fine-grained senses, assess its quality and show that it helps improve sense disambiguation on standard test sets.

[2] `http://wordnet.princeton.edu/man/wnstats.7WN.html`

1.4 Related Work

Systems using WordNet as the underlying ontology often suffer because of the fine-grained nature of the sense inventory. With more and more applications using WordNet, clustering word senses such that the application developer has the control over the granularity becomes very important.

One of the earliest attempt at coarsening of Machine Readable dictionaries was made by [**?**]. They tried to discover similarities between senses of Longman's Dictionary of Contemporary English(LDOCE) using multiple heuristics based on a variety of information about a sense's meaning.

A wide variety of automatic methods have been proposed since then for coarsening fine-grained inventories. In this section, we discuss the approaches proposed in literature to coarsen sense inventories. The different ideas to cluster senses can be broadly categorized as follows:

- Merging senses based on the sense ontology structure.
- Clustering based on word sense similarity estimated from external corpora.
- Exploiting Disagreements: between human annotators or between WSD systems.
- Using translational equivalences of word senses.
- Mapping to coarser sense inventories.
- Clustering senses using supervision.

1.4.1 Merging senses based on ontology structure

[**?**] suggest merging of two word senses based on structural cues derived from the ontology structure. They cluster senses which are connected by ontology based

relations like *twins*[3], *autohyponymy*[4], *sisters*[5], *cousins*[6] etc. [**?**] extended this idea and proposed six semantic principles to merge synsets and probabilistic principles to drop infrequent synsets. Some interesting principles include merging synsets sharing a *pertainym*, *antonym* or sharing the same *verb group*[7]. This was the first attempt to group synsets instead of word senses.

A number of synset similarity measures based on the WordNet structure have also been proposed in literature like Path Based Similarity Measures by [**?**], [**?**], Information Content Based Measures by [**?**], [**?**], [**?**] and Gloss Based Heuristics by [**?**],[**?**]. We discuss these similarity measures in detail in Section 3.4.1.1. Though these measures have not been used directly for WordNet sense clustering, they have motivated researchers in the NLP community to make full use of the WordNet structure to capture similarity between senses.

1.4.2 Clustering based on Word Sense similarity estimated from External Corpora

For estimating similarity between words, many corpus oriented attempts have been made like [**?**], [**?**], [**?**] and [**?**]. The problem with these approaches is that they are not able to handle the polysemous nature of words. Also, the dearth of sense annotated corpora prevents most of these methods to be used effectively in computing word sense similarities.

[**?**] associated a topical vector with each word sense, called *Topic Signature*, which captures the relatedness between the target word sense and vocabulary. The cosine similarity between the vectors of two senses is used as the similarity between them. To calculate these vectors, they collect contexts for a polysemous words

[3]Two synsets having more than one word common.
[4]If one sense is a direct descendant of the other in the hypernym-hyponym ontology.
[5]Word senses that share the same hypernym
[6]Node pairs whose hyponyms exhibit a specific relation to each other: identified and listed by lexicographers in WordNet 1.5.
[7]Verb Groups are manually determined by lexicographers.

from manually sense-tagged corpora and by using instances of a polysemous word's monosemous relatives[8] from large untagged corpora and the web. The idea behind the approach is that because of lack of sense annotated data, semantically close senses may not have a lot of direct shared contexts, but they are expected to have similar distributional neighbours.

On similar lines is the approach by [**?**]. They calculate sense similarities using a combination of the JCN WordNet based similarity measure [**?**] and word-to-word distributional similarity. They introduce a more relaxed notion of sense relatedness which allows the user to control the granularity for the application in hand.

1.4.3 Exploiting Disagreements: between WSD Systems or between Human Annotators

The central idea involved here is that whenever good WSD systems or human annotators get confused while disambiguating, the senses they mark as answers are semantically related to the correct answer.

Example 1. Consider the following noun senses of the word *fish*, taken from WordNet 3.1:

- fish (any of various mostly cold-blooded aquatic vertebrates usually having scales and breathing through gills) "the shark is a large fish"
- fish (the flesh of fish used as food) "in Japan most fish is eaten raw"
- Pisces, Fish ((astrology) a person who is born while the sun is in Pisces)
- Pisces, Pisces the Fishes, Fish (the twelfth sign of the zodiac; the sun is in this sign from about February 19 to March 20)

It is unlikely that a human annotator mistags the astrology sense of the word *fish* with its food sense. Similar results are expected from a good WSD system as well.

[8] Single-sense synsets related by hyponym, hypernym or any other relation of WordNet.

[?] derives confusion matrices exploiting the disagreements between human annotators and uses the same to generate coarse sense clusters. On the other hand, [?] uses the publicly available outputs of Senseval-2 [?] participants to construct the confusion matrices between word senses and then cluster them using hierarchial agglomerative clustering. Dearth of available manually sense-tagged data and not-so-high performance of the WSD systems limits the applicability of the above mentioned ideas.

1.4.4 Using translational equivalences of word senses

[?] constructed similarity matrices for Senseval-2 [?] words based on translations of these words in four languages. The principle involved can be summed as: *two word senses which are translated with the same word sense in other language are expected to be semantically similar*. Using multiple languages enables us to cover all the sense divisions offered by different languages. [?] uses the similarity matrices provided by [?] and report resulting hierarchial clusters.

With the advent of WordNets being developed in multiple languages[9] as well as multilingual ontologies like BabelNet [?], this seems a promising area which can help in coarsening of senses.

1.4.5 Mapping to coarser sense inventories

Mapping WordNet to other inventories either manually or automatically to generate coarse senses has also been tried by many researchers in the NLP community. When the different WordNet senses map to the same sense in the other ontology via manual mapping or automatic mapping, it is expected that the senses must have been semantically close. The underlying assumption being that the automatic mapping is able to capture the semantic similarity between the concepts in both the ontologies with high efficacy.

[9]GlobalWordNet lists the WordNets available in the public domains: `http://www.globalwordnet.org/gwa/wordnet_table.html`.

The attempts made in this direction include mapping WordNet to Hector Lexicon [?], PropBank [?] and Levin Classes [?] [?]. Most of these mappings are not complete in both directions, which hampers their utility.

The automatic approach presented by [?] for mapping between sense inventories, WordNet to Oxford English Dictionary to be precise, is an elegant approach exploiting similarities in glosses and semantic relationships in the sense inventories. The approach can be extended to discover more semantic relationships in WordNet by using the ontology structure of the ontology mapped to WordNet.

1.4.6 Clustering senses using supervision

One of the earliest attempt to cluster senses using supervision was proposed by [?]. They train a Support Vector Machine [?] using features derived from WordNet and other lexical resources, whose predictions serve as a distance measure between synsets. For synset pairs with no common words, they assume a zero sense similarity score. They cluster synsets using average link agglomerative clustering and the synset similarity model learnt. While merging synsets to construct a coarse taxonomy, the merged synset inherits its hypernymy from its most frequent constituent sense. All other relationships are added to the new merged sense as long as the acyclic nature of the relations is conserved.

1.5 Discussion

In this section, we discuss some ideas which shaped the direction of our approach to the problem of WordNet sense clustering.

1.5.1 Dropping Infrequent WordNet Synsets

To reduce polysemy in WordNet, [?] proposed to drop infrequent synsets, along with merging similar senses. They scored the synsets using frequency information of senses in SemCor [?] and dropped the low scoring synsets.

We study the usefulness of this idea on the SemEval 2007 WSD test datasets [**?**], in which we have three generic Wall Street Journal texts and two domain-specific documents (relating to computer programming and Italian painting respectively). Our scoring function is similar to the one proposed by [**?**] and is given by:

$$Score(Synset) = \sum_{i \in Synset} Score(WordSense_i) \tag{1.1}$$

$$Score(WordSense) = \frac{Frequency(WordSense) + \alpha}{Frequency(LemmaWord) + \alpha * NumberOfSenses(LemmaWord)} \tag{1.2}$$

Here α is the Laplacian correction parameter(for our study we take $\alpha = 1$). The scoring function does not make reference to the component word senses directly and thus we do not have to deal with the data sparseness issue that arises from the limited size of the corpus.

We drop the synsets whose score is less than a threshold and study the number of instances for which the correct answer was removed, thus accounting for error. Since the task was on coarse grained WSD evaluation, we have multiple senses as the correct answer. In this case, we measure the fraction of senses removed from the set of correct senses for each of the instances. We report the error score calculated, weighted by the number of instances in the documents.

		Threshold		
		0.03	0.06	0.1
Corpora	WSJ	7.72	13.91	18.77
	Domain	10.48	14.19	19.13

Table 1.2: Dropping Infrequent Synsets - A Comparision

We observe that for lower thresholds, the error rate is higher for domain specific datasets whereas as we increase the threshold, thus removing more and more synsets, the error rates of both the datasets become comparable. The results match

our expectation as we expect that some of the words in domain specific datasets would have low frequencies and thus low scores. To summarise, while working with text from the general domain, removing some infrequent synsets might improve the disambiguation but in case of domain specific datasets, it should not be done.

1.5.2 Clustering Synsets Vs Clustering Senses

For generating a coarse sense inventory, many researchers have focused on clustering WordNet senses into groups i.e. generate coarse senses for each word by merging its senses [**?**] [**?**] [**?**]. In this approach, we would like to highlight two problems. One problem is that it requires a stopping criterion for each word such as the number of final classes. In literature, researchers have used the numbers determined by the available gold standard datasets for the purposes of evaluation [**?**]. As the right number of classes for each word cannot usually be predetermined even if the application is known, such coarsening systems cannot be used to derive coarse senses for all the words. The other problem is the inconsistent sense clusters obtained because of independent generation of coarse senses for each word. Even manually done sense clusterings can have this error: consider the sense groupings of the verbs textitneed and *require* from Senseval-2 judgements in figure 1.2 [**?**].

The two WordNet 2.1 senses *need#v#1* and *need#v#2* clustered in word specific labelling for *require*, are not clustered in word specific labelling for *need*. These transitive closure errors suggest that for deriving consistent coarse senses, we should cluster synsets and not senses.

1.6 Evolution of Evaluation Frameworks

We would like to highlight here the different frameworks used in literature for evaluating the sense clustering problem. Early systems studied the quality of clusters of word senses by studying polysemy degree of the text [**?**] or by measuring the entropy and purity of the clusters obtained [**?**].

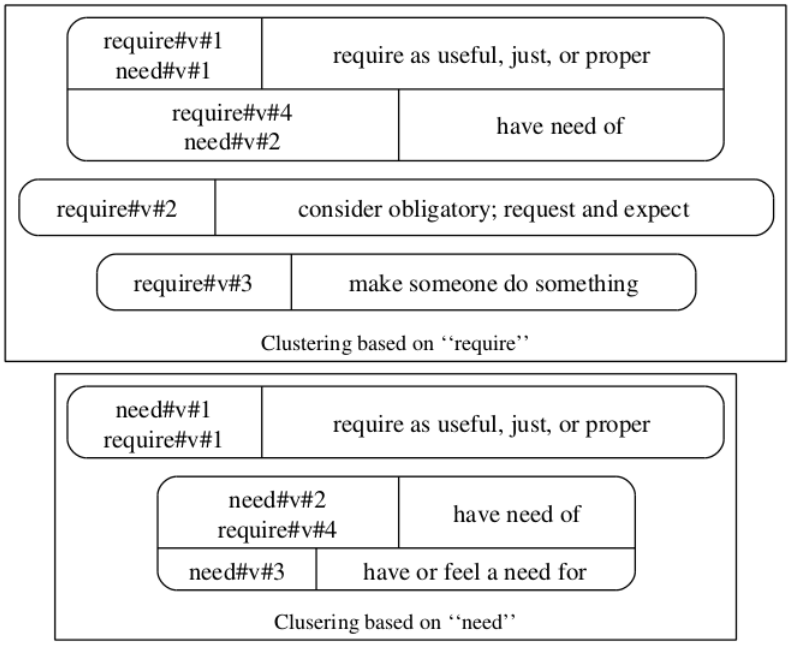


Figure 1.2: Inconsistency in sense groupings for the verbs *need* and *require* from Senseval-2 judgements

Another idea is to compare the clustering obtained against a manually sense clustered dataset as done by [**?**]. This can be done by treating clustering as a pairwise classification task and reporting the F-Score for the classification task. The problem with this approach of evaluation is that the number of intra-cluster pairs is often small compared to inter-cluster pairs in clustering tasks. This imbalance could lead to understating pairwise similarity and can be avoided by using FScore for both the classes as performance evaluators.

The recent line of thought in evaluation is to go for a task based evaluation. [**?**] studied the performance of first sense heuristics in the Senseval-2 English LS Task [**?**]. [**?**] and [**?**] assess the effect of automatic sense clustering on the three best-ranking WSD systems of the English AW task at Senseval-3 [**?**]. Since the main motive for coarsening WordNet inventory is to make WSD a plausible task, studying the performance of sense disambiguation systems on automatically generated coarsened senses seems a well founded approach.

1.7 Broad Overview of our Approach

We propose a framework which derives a coarse sense inventory by learning a synset similarity metric. We focus on the coarsening of noun synsets of WordNet and show that the coarse-grained sense inventory obtained notably improves the disambiguation of nouns.

Our approach closely resembles [**?**] as far as supervised learning of the synset similarity is concerned. But to learn synset similarity of synset pairs which don't share a word, instead of giving them zero similarity, we learn it using a variant of the SimRank framework [**?**].

Also, [**?**] proposes to modify the WordNet ontology structure to produce a coarse version of WordNet; however we argue on the lines of [**?**] that a softer notion of relationships between senses should be used compared to fixed groupings so that the variable granularity needs of different applications can be met.

1.8 Organization of Thesis

In this chapter, we introduced the problem, reviewed the algorithms proposed for the task of sense clustering and the evaluation frameworks for the same. The rest of this thesis is organized as follows. Chapter 3 discusses a supervised attempt to capture WordNet synset similarity using various features derived from WordNet and external corpora. Chapter 4, presents a semi-supervised approach to estimate WordNet synset similarity using a variant of SimRank [**?**] and describes our approach to produce a coarse sense inventory. The conclusions and future work are part of chapter 5.

Chapter 2

Related Work

The task of text classification, i.e. classification of documents into a fixed number of predefined categories has been long studied in-depth for many years now. This multi-class classification problem has further evolved into a multi-label text classification task where each document can belong to multiple, exactly one or no category at all.

Supervised machine learning techniques that learn classifiers to perform this category assignment task can be broken down into two main components, namely, text representation and learning algorithm. Text representation involves converting the documents, that are usually strings of characters, into numerical vectors that are suitable inputs to the learning algorithm while the learning algorithm uses pairs of labeled input text representations and the categories it is belongs in, to learn a model so as to classify new documents into categories.

2.1 Text Representation

Any text-based classification system requires the documents to be represented in an appropriate manner dictacted by the task being performed [Lewis, 1992]. Moreover, [Quinlan, 1983] showed that the accuracy of the classification task depends as much on the document representation as on the learning algorithm being employed. Different from the data mining task, which deals with structureed documents, text classification deals with unstructured documents that need to be appropriately trans-

formed into numerical vectors, i.e. the need for text representation. In this section we introduce the most effective and widely-used techniques to represent documents for text classification.

2.1.1 Bag of Words

It is found in information retrieval research that word stems work well as representations units for documents and that their ordering in a document is of minor importance for many tasks. This is attributed by the fact that the most widely-used used model to represent documents for the classification task is the *Vector Space Model (VSM)* [Salton and Yang, 1973].

In the Vector Space Model, a document d is represented as a vector in the term/word space, $d = (w_1, w_2, \ldots, w_{|V|})$ where $|V|$ is the size of the vocabulary. Each of the $w_i \in [0, 1]$, represents the weightage of the term i in the document d. This is called the *bag-of-words* model as it ignores word ordering and each document is reduced to a bag of words that it contains or not.

An important requirement of such a representation is that, the terms that help in defining the semantic content of the document and play an important role in classification be given higher weightage than the others. Over the years, there has been much research in the information retrieval field on term weighting schemes. The most important term-weighting techniques are described below :

1. **One Hot Representation** : This is the most trivial representation, where each document is represented by a vector that is size of the vocabulary. Each element in the vector is either a 0 or a 1 to denote the absence or presence of a specific term in the document.

2. **Term Frequency (tf))** : The term frequency representation weighs the terms present in the document relative to their occurence frequency in the document. Hence a document d is represented as, $d = (w_1, w_2, \ldots, w_{|V|})$, where, w_k is the number of times the term k appears in the document d.

3. **Inverse Document Frequency (idf)** : Though using *tf* as a term weighting scheme is a good starting point, it faces a challenge when high frequency terms are not concentrated in a few particular documents but are prevalent in the whole collection. Those terms then stop being characteristic of the semantic content of a few documents and need not be given high weightage. To overcome this problem, Salton and Buckley [1988] suggested a new term weighting called the inverse document frequency (idf). The *idf* weight of a term varies inversely with the number of documents n it belongs to in a collection of total N documents. A typical *idf* vector can be computed as

$$w_k = \log \frac{N}{n} \tag{2.1}$$

4. **Term Frequency Inverse Document Frequency (tf-idf)** : Given the above two term weighing schemes, it is clear that an important term in a document should have high *tf* but a low overall collection frequency (*idf*). This suggests that a reasonable measure for term importance may be then obtained by the *tf* and the *idf* ($tf \times idf$). As we will see in the results section, the *tf-idf* weighed bag-of-words document representation gives one of the best accuracies in the multi-label text classification task.

A common feature in the bag-of-words document representation is the *normalization factor*[Salton and Buckley, 1988] introduced to reduce the effect of varying document lengths and give equal weightage to documents of all lengths when learning the classifier for text categorization. **TODO:** Do we put how normalization is done? Another feature added to the bag-of-words representation is the removal of stop-words (short function words that do not add to the semantic content of the document) and words that occur infrequently to make the document vector more meaningful.

2.1.2 Dimensionality Reduction / Feature Selection

TODO: Why need feature selection? Different techniques for feature selection. IG, Chi Sq., MI, LSI The bag-of-words representation scheme has several drawbacks but the most important drawback it suffers from is that document vectors are very sparse and high dimensional. Typical vocabulary sizes of a moderate-sized document collection ranges from tens to hundereds of thousands of terms which is prohibitively high for many learning algorithms. To overcome this issue of high-dimensional bag-of-words document representations, automatic feature selection is performed that removes uninformative terms according to corpus statistics and constructs new orthogonal features by combining several lower level features (terms/words). Several techniques used in practice are discussed below,

1. **Information Gain** : Information Gain is widely used as a term-goodness criterion in the field of machine learning, mainly in decision trees [Quinlan, 1986] and also in text classification [Lewis and Ringuette, 1994], [Moulinier et al., 1996]. It is a feature space pruning technique that measures the number of bits of information obtained(entropy) for category prediction by knowing the presence or absence of a term in a document. For terms where the information gain was below some predefined threshold are not considered in the document vector representation. The information gain of a term t is defined as

$$G(t) = -\sum_{i=1}^{|C|} P(c_i) \log P(c_i) + P(t) \sum_{i=1}^{|C|} P(c_i|t) \log P(c_i|t) + P(\ t) \sum_{i=1}^{|C|} P(c_i|\ t) \log P(c_i|\ t) \tag{2.2}$$

2. **Mutual Information** : Similar to the Information Gain scheme, Mutual Information estimates the information shared between a term and a category and prunes terms that are below a specific threshold. The mutual information between a term t and a category c is estimated in the following fashion,

$$I(t, c) = \log \frac{P(t \wedge c)}{P(t) \times P(c)} \tag{2.3}$$

To measure the goodness of a term in global feature selection, the category specific scores of a term are combined using,

$$I_{avg}(t) = \sum_{i=1}^{|C|} P(c_i) I(t, c_i) \tag{2.4}$$

3. χ^2 **Statistic** : The χ^2 statistic measures the lack of independence a term t and a category c and can be compared to the χ^2 distribution with one degree of freedom. The term-goodness factor is calculated for each term-category pair and is averaged as above. The major difference between Mutual Information and χ^2 statistic is that the later is a normalized value and the goodness factors across terms are comparable for the same category.

4. **Latent Semantic Indexing (LSI)** :

Chapter 3

Supervised Synset Similarity

In this chapter, we describe a supervised attempt at learning synset similarity in which we train a Support Vector Machine using a semantically rich collection of features obtained from the WordNet ontology and other publicly available lexical resources connected to WordNet.

3.1 Motivation

Most of the synset similarity measures, like JCN [**?**], LCH [**?**] etc., proposed in literature are generally unsupervised, relying mostly on WordNet structure and raw text corpora. Our aim in this chapter is to provide a supervised alternative to the above mentioned approaches, which utilizes the WordNet structure to its fullest and makes use of additional resources like WordNet Domains [**?**], SemCor [**?**], SentiWordNet [**?**] etc as well.

Supervised systems allow us to intelligently combine and weigh the different features and thus give us an insight into how humans relate word senses.

3.2 Algorithm Outline

We formulate sense merging as a binary classification problem and tackle the same using supervision. We obtain pairs of synsets which human-annotators have labeled

as “merged” or “not merged” and describe each pair as a feature vector. We learn a synset similarity measure by using a maximum margin classifier on this extracted dataset, where positive examples are the pairs which were merged and negative examples are the ones which were not merged by the annotators. The output value of the classifier for a synset pair is the learnt estimate of the similarity between synsets constituting the pair.

3.3 Gold standard sense clustering data

In this section, we will discuss the gold standard sense clustering datasets, the preparation of the pairwise classification datasets from them and the quality of these datasets.

Since our methodology depends upon the availability of labelled judgements of synset relatedness, the datasets involved are of paramount importance. We use the following manually labelled WordNet sense groupings:

- Sense groupings over WordNet senses provided by the Senseval-2 English LS task organizers [**?**].
- Mappings from the Omega ontology [**?**] to the WordNet senses, provided by the OntoNotes project [**?**]

3.3.1 Senseval-2 Dataset

Hand labelled sense clusters on selected nouns, verbs and adjectives was provided by the Senseval-2 English LS Task on WordNet 1.7 [**?**] [**?**].

Structure of dataset : Let us consider the sense groupings of the noun *air* given below. Annotator(s) have divided 8 senses of the word *air* in 5 groups. The first group consists of 4 senses and the remaining groups consists of one sense each.

```
air%1:15:00:: 4 air%1:27:00::
air%1:19:00:: 4 air%1:27:00::
```

air%1:27:01:: 4 air%1:27:00::

air%1:04:00::

air%1:10:02::

air%1:07:00::

air%1:10:01::

Binary Classification Data Generation : From these clusters we generate a binary classification dataset where the task in hand is to label a pair to be merged or not. The process of extracting pairs is as follows:

- Part of speech based separation of the word senses was done for convenience. Thus we obtained clusters of senses for each POS (noun, verb, adjective and adverb).
- We map all these word senses to WordNet 3.0 synsets and get the clusters over synsets.[1]
- Among all possible pairs from the obtained synsets, if both the offsets are present in the cluster, we label the pair as positive/merged, negative/not-merged otherwise.

Statistics : The statistics related to the pairs obtained is tabulated in table 3.1. There was no pair for any of the POS which was in negative examples as well as positive examples.

POS	Positive example count	Negative example count	Fraction of positive examples
Noun	119	552	0.22
Verb	477	4769	0.10

Table 3.1: Senseval-2 Pairwise Data Statistics

[1]We use the mappings provided by the WordNet project and provided by Dr. Rada Mihalcea http://www.cse.unt.edu/~rada/downloads.html#wordnet

Quality of Dataset : Unfortunately, no information about the dataset construction is known nor any inter-annotator agreement is reported in literature.

3.3.2 OntoNotes Dataset

We use OntoNotes [**?**] Release 3.0 [2] for extracting WordNet sense clusters.[3]

Structure of Dataset : The dataset consists of senses for selected words in sense files. The senses in OntoNotes are mapped to WordNet senses, if a good mapping between senses exists.

Binary Classification Data Generation : The steps involved in extraction are as follows:

1. Simplifying Dataset: We extracted the relevant portions of the sense inventories i.e. the mapping from OntoNotes senses to WordNet senses.

2. Version wise separation:

   - OntoNotes has mappings to 4 WordNet versions: 1.7, 2.0, 2.1 and 3.0.
   - We partitioned the inventory according to the versions.
   - We manually resolved the cases containing multiple versions of wordnet in a single sense file.
   - We dropped monosemous words as there is no clustering possible in such cases.

3. Mapping senses to WordNet 3.0:

   - We dropped WN1.7 as there were very few senses and the mapping from WN1.7 to WN3.0 was not easily available. [4]

4. Validating clusters on WN3.0:

[2] `http://www.ldc.upenn.edu/Catalog/docs/LDC2009T24/OntoNotes-Release-3.0.pdf`
[3] The OntoNotes groupings will be available through the LDC at `http://www.ldc.upenn.edu`
[4] Mapping could have been obtained by using following mappings: WN1.7 to WN1.7.1 , WN1.7.1 to WN2.0, WN2.0 to WN2.1 and then WN2.1 to WN3.0.

- We removed the sense files which did not contain all the senses of the word i.e. the clustering was not complete.
- We removed the sense files in which the clusters had a clash i.e. one sense belonged to multiple clusters.
- We removed the sense files in which there were invalid offset mappings.

5. Removing Disagreements: We removed the instances from the dataset which were present in both positive and negative examples. This situation arises because the annotators were working with word senses and there were inconsistent sense clusters.

Statistics :

- **Number of Sense Files**: Effect of processing on number of sense files

Stage	Noun	Verb
Before Processing	2033	2156
After processing	1680	1951

Table 3.2: Effect of processing on sense files

- **Average Polysemy Degree**: Average Polysemy Degree is defined as the average polysemy degree of the words weighted by their frequency in a large corpus. The equation 3.1 defines the same [5].

$$AvgPolyDeg = \frac{\sum_i polysemyDegree(word_i) * freq(word_i)}{\sum_i freq(word_i)} \tag{3.1}$$

Average Polysemy degree of words in the cleaned dataset:

Stage	Noun	Verb
Fine	5.28	9.55

[5]Frequency of a word is the sum of the frequencies of all its sense occurrences, which we obtain from WN3.0.

Coarse	4.10	3.97

Table 3.3: Average Polysemy Degree of Cleaned OntoNotes dataset

For comparison purposes, we would like to report the average polysemy degree in the dataset released for Semeval-2007 task [**?**] on coarse grained WSD. It is based on Wordnet 2.1 and was created automatically as described in [**?**]. The dataset consists of only polysemous entries of which there are 11052 nouns and 4121 verbs.

Stage	Noun	Verb
Fine	3.76	5.84
Coarse	2.18	2.54

Table 3.4: Average Polysemy Degree of SemEval-2007 clustering

Binary Classification Dataset From the word sense clusters, we obtained the pairwise classification data for both nouns and verbs separately using the method described in 3.3.2.

Statistics	Nouns	Verbs
Number of Word Sense Files	1680	1951
Distinct Offsets encountered	4930	6296
Positive Examples	1214	6881
Negative Examples	11974	20899
Percentage of Positive examples	9.20	24.76

Table 3.5: Statistics of Pairwise Classification Dataset obtained from OntoNotes

Quality of Dataset : The OntoNotes dataset creation involved a rigourous annotation process, in which around 50 examples for every target noun or verb were tagged with a preliminary set of senses. This process was continued until an ITA of atleast 90% was reached, and in each iteration, the sense groupings were revised

by the linguists and the examples were tagged again. Thus, using the coarse sense inventory of the final iteration, an ITA of at least 90% is guaranteed on the sense-tagging of the sample sentences, which makes the quality of the final clustering of senses reasonably high. The process is summarised in figure 3.1.

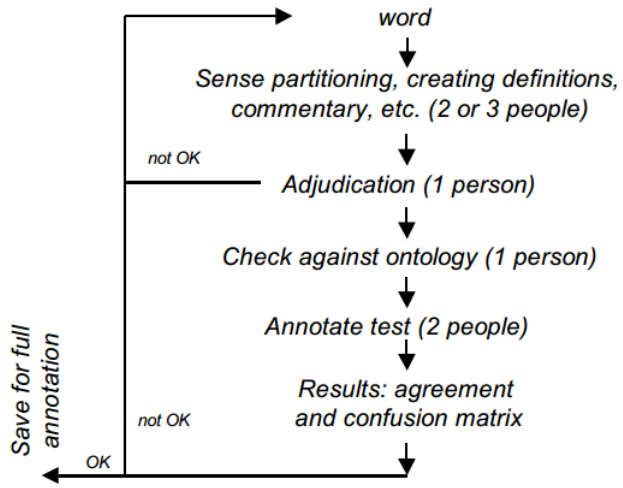


Figure 3.1: OntoNotes Annotation Procedure [?]

3.4 Feature Engineering

In this section, we describe the feature space construction. We derive features from the structure of WordNet and other available lexical resources. Our features can be broadly categorized into two parts: derived from WordNet and derived from other corpora. WordNet based features are further subdivided into similarity measures and features. Many of the listed features are motivated by [?] and [?].

3.4.1 WordNet based Features

3.4.1.1 Similarity Measures

1. **Wu and Palmer's Conceptual Similarity**: WUP Similarity [**?**] between concepts a and b in a hierarchy, given as

$$sim_{WUP}(a,b) = \frac{2 \times depth(lso(a,b))}{len(a,lso(a,b)) + len(b,lso(a,b)) + 2 \times depth(lso(a,b))} \tag{3.2}$$

   Here $depth(lso(a,b))$ is the global depth of the lowest super ordinate of a and b and $len(a,b)$ is the length of the path between the two nodes a and b in the hierarchy.

2. **Leacock and Chodorow's Normalized Path Length**: LCH Similarity [**?**] between concepts a and b in a hierarchy is given by:

$$sim_{LCH}(a,b) = -\log\left(\frac{len(a,b)}{2 \times \max\limits_{c \in WordNet} depth(c)}\right) \tag{3.3}$$

3. **Resnik's Information Theory Based Approach**: Information Content for a concept C in the taxonomy is defined as $-\log_2 p(c)$, where $p(c)$ is the probability of encountering an instance of concept c. Resnik's similarity, introduced in [**?**], is given by

$$sim_{RES}(a,b) = -\log p(lso(a,b)) \tag{3.4}$$

   Observe that in a taxonomy with a unique top node, the similarity between a pair of concepts having the top most node as their most-specific subsumer will be 0.

4. **Jiang and Conrath's Combined Approach**: JCN distance [**?**] between

concepts a and b is given by:

$$dist_{JCN}(a,b) = IC(a) + IC(b) - 2 \times IC(lso(a,b)) \tag{3.5}$$

$$= 2\log p(lso(a,b)) - (\log p(a) + \log p(b)) \tag{3.6}$$

We use the inverse of JCN distance as a feature, which we call JCN similarity.

5. **Lin's Universal Similarity Measure**: Lin's measure of similarity between two concepts a and b in a hierarchy, proposed in [**?**], is given by :

$$sim_{LIN}(a,b) = \frac{2 \times \log p(lso(a,b))}{\log p(a) + \log p(b)} \tag{3.7}$$

6. **Lesk Variants**: We use the following variants of Lesk Similarity [**?**] as features: Adapted Lesk[**?**], Adapted Lesk Tanimoto and Adapted Lesk Tanimoto without hyponyms. The Adapted Lesk Tanimoto is a variant of Adapted Lesk, where we use the glosses of all the directly related senses and the hyponyms of the hypernym of the sense as well. Jaccard-Tanimoto Coefficient is then calculated over two vectors containing the count of each word in the expanded gloss. The Adapted Lesk Tanimoto No Hyponyms is a simpler version of Adapted Lesk Tanimoto in which we only use direct WordNet relations.

We illustrate the similarity values obtained using above measures in table 3.6 on the following pairs of synsets:

- {*student#n#1, pupil#n#1, educatee#n#1*: a learner who is enrolled in an educational institution} and {*student#n#2, scholarly_person#n#1, scholar#n#1, bookman#n#1* : a learned person (especially in the humanities); someone who by long study has gained mastery in one or more disciplines}
- {*student#n#1, pupil#n#1, educatee#n#1*: a learner who is enrolled in an educational institution} and {*teacher#n#1, instructor#n#1* : a person whose occupation is teaching }

Similarity Measure	Pair 1	Pair 2
LCH	2.0794	1.7429
WUP	0.8	0.7272
JCN	0.0984	0.0905
LIN	0.2725	0.2562
RES	1.9033	1.9033
AdapLesk	530.0	170.0
AdapLeskTani	0.1459	0.0696
AdapLeskTaniNoHypo	0.1742	0.1471

Table 3.6: Similarity Measures Illustration

3.4.1.2 Features

1. **Common word lemmas count**: Number of lemmas common in two synsets

2. **SenseCount**: maximum polysemy degree among the lemmas shared by the synsets

$$\max_{lemma \in Synset_1 \cap Synset_2} \text{Number of Senses}(lemma)$$

3. **SenseNum**: number of lemmas having maximum polysemy degree among the lemmas shared by the synsets

$$\left| \arg\max_{lemma \in Synset_1 \cap Synset_2} \text{Number of Senses}(lemma) \right|$$

4. **Same lexicographer file**: Synsets in WordNet are divided in several broad categories by the lexicographers. For more details visit Appendix A. We check whether two synsets have the same category or not.

5. **SP1_1 merge heuristic** [**?**]: Binary feature which is set if S1 and S2 are two synsets containing at least 2 words, and if S1 and S2 contain the same words.

6. **SP1_2 merge heuristics** [**?**]: Binary feature which is set if S1 and S2 are

two synsets with the same hypernym and contain the same words. The strict heuristic checks whether all the hypernyms are shared or not whereas the relaxed heuristic checks if the synsets have at least 1 common hypernym.

7. **SP1_3 merge heuristic** [**?**]: Binary feature which is set if S1 and S2 are two synsets with at least K words in common. We set $K = 3$ here, which makes it equivalent to the *twin relation*.

8. **Number of common hypernyms**: the number of common hypernyms between the two synsets.

9. **Autohyponymy**: Whether the two synsets have a hyponym-hypernym relation between them.

3.4.2 Features derived from other Corpora

3.4.2.1 eXtended WordNet Domains

eXtended WordNet Domains Project [**?**] provides us the score of a synset with respect to a domain-label. The dataset contains 169 labels(excluding factotum label) which are hierarchically organized for different POS. We obtain a representation of a synset in the domain label space and use cosine similarity, L1 Distance and L2 Distance computed over the weight representations of the synsets as features.

3.4.2.2 BabelNet

BabelNet [**?**] provides us with two very important datasets of usage. One of them is the translation of noun word senses in 6 languages namely: English, German, Spanish, Catalan, Italian and French. Secondly mapping of noun synsets to DBpedia entries. For features we use counts of common lemmas in all 6 languages and count of common dbPedia entries.

The idea to cluster word senses using translation equivalences was proposed first in [**?**].

3.4.2.3 SentiWordNet

SentiWordNet [**?**] provides us with a mapping from a synset to a triad of three weights. The weights correspond to the score given to a synset based on its objectivity and subjectivity(positive and negative). For eg.

- The synset {*sprightliness#n#1, spirit#n#7, liveliness#n#2, life#n#9*: animation and energy in action or expression; "it was a heavy play and the actors tried in vain to give life to it" } has the score (P: 0.25, O: 0.75, N:0)

We use cosine similarity, L1 Distance and L2 Distance of the weight representations of the synsets as features.

3.4.2.4 Mapping of WordNet to Oxford English Dictionary(OED)

We derive one of our feature from the sense clusterings produced by mapping WordNet senses to OED senses using Structural Semantic Interconnections method [**?**] by organizers of coarse-grained AW task in SemEval-2007[6] [**?**]. For each pair of synsets, we check if there are senses in the synsets such they belong to same cluster in the OED mapping.

3.5 Classifier and Training

We train Support Vector Machines using features described in section 3.4 on the dataset of synset pairs mentioned in section 3.3, in which every synset pair is given either a "merged" or "not-merged" label. We will now discuss Support Vector Machines and various normalization techniques and kernels, for completeness.

3.5.1 Support Vector Machines

Support Vector Machines, proposed by [**?**], are linear machines which try to separate instances of opposite classes by constructing an optimal hyperplane having

[6] `http://lcl.uniroma1.it/coarse-grained-aw/`

the largest possible margin. They are widely used for classification tasks and are known for their high accuracies and robustness. An example of such hyperplane is illustrated in figure 3.2.

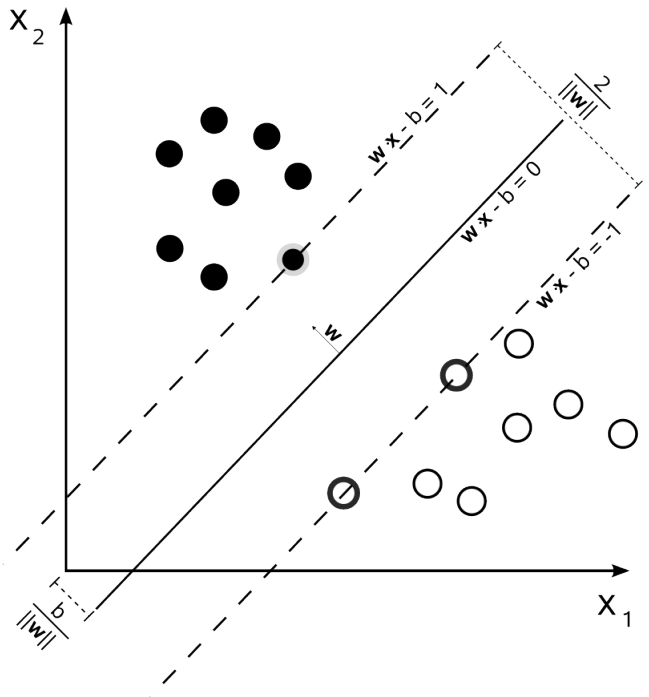


Figure 3.2: SVM with maximum margin hyperplane (Source: Wikipedia)

In many applications, non-linear classifiers provide better accuracies than linear classifiers as they are able to discover better decision boundaries. But linear classifiers have advantages over non-linear classifiers as they often have simple training algorithms that scale well with the number of examples. [**?**] introduced the idea of kernels which use the machinery of linear classifiers to discover non-linear decision boundaries by fitting the maximum-margin hyperplane in a transformed feature space. Some common kernels include:

- Polynomial: $k(\boldsymbol{x_i}, \boldsymbol{x_j}) = (\boldsymbol{x_i} \cdot \boldsymbol{x_j})^d$

- Gaussian Radial Basis Function: $k(\boldsymbol{x_i}, \boldsymbol{x_j}) = exp(-\gamma||\boldsymbol{x_i} - \boldsymbol{x_j}||^2)$

We experiment with the standard linear kernel and the RBF kernel.

3.5.2 Feature Normalization

Feature Normalization plays a very important role in many learning and optimization algorithms. In practice, many methods work best after the data has been normalized and whitened. Since Support Vector Machine algorithms are sensitive to scaling and have been shown to give better results with normalization, we experiment with two ideas - feature scaling and feature standardization:

3.5.2.1 Feature Scaling

One of the simplest methods to normalize features is to scale the ranges of features to a common range, [-1,1] in our case. The main advantage offered by scaling is that it avoids attributes in smaller numeric ranges being dominated by attributes in greater numeric ranges. It also avoids numerical difficulties like overflow errors caused by large attribute values. Note that both training and testing data should be scaled with the same parameters, otherwise the results would be erroneous. The transformation is obtained by:

$$x' = minReqd + (maxReqd - minReqd) * \left(\frac{x - min}{max - min}\right) \tag{3.8}$$

3.5.2.2 Feature Standardization

For a heterogeneous feature space, feature standardization makes the values of each feature in the data have zero-mean and unit-variance. The following transformation formula achieves the zero-mean and unit-variance requirements:

$$x' = \frac{x - \mu}{\sigma} \tag{3.9}$$

The mean(μ) and variance(σ^2) used in equation 3.9 are the sample mean and unbiased sample variance, estimated from the training data. Note that the same transformation must be applied to train and test data to obtain meaningful results.

3.6 Implementation

In this section, we outline some of the important resources which have been used in the implementation:

- Access to the WordNet graph was facilitated by extJWNL [7].
- For WordNet based Word Similarity, we make use of Java WordNet::Similarity [8] by David Hope, which is a pure Java implementation of Ted Pedersen's Perl WordNet::Similarity [9].
- The BabelNet based features were obtained using the BabelNet API [**?**].
- To train the support vector machine classifier we used SVM implementation by [**?**], whose java access is provided by JNI-SVMLight [10] library.
- For Information Gain and Gain Ratio study in section 3.7.4.1, we use Weka software [**?**].

3.7 Experimental Setup and Evaluation

3.7.1 Train and Test datasets

Since the quality assurance of Ontonotes dataset is reasonably high and no information is available about Senseval-2 dataset, we use binary classification dataset obtained from OntoNotes for training and validation. We split our dataset into a training set(70%) and a held-out validation set(30%).

Examples	Nouns
Positive Examples[11]	1214

[7] `http://extjwnl.sourceforge.net/`
[8] `http://www.sussex.ac.uk/Users/drh21/`
[9] `http://wn-similarity.sourceforge.net/`
[10] JNI-SVMLight: `http://adrem.ua.ac.be/~tmartin/`
[11] Pair of synsets merged by annotators

Negative Examples[12]	11974
Percentage of Positive examples	9.20
Positive Training examples in random 70% sample	850
Negative Training examples in random 70% sample	8382
Positive Testing examples in random 30% sample	364
Negative Testing examples in random 30% sample	3612

Table 3.7: Statistics of Pairwise Classification Dataset

3.7.2 Effect of class distribution in learning

We trained two systems, which differ in number of negative examples used in training. One uses the whole 70% dataset extracted and other selects random instances from negative examples of this dataset to get a balanced dataset (equal number of positive and negative instances). For testing, we again used a balanced dataset consisting of equal number of positive and negative instances, disjoint from the training dataset. The former classified all the instances into negative class. We attribute this to the skewed class distribution in the training data. On the other hand, we observe that system 2 learns better boundaries due to the balanced nature of the training set.

Owing to the above observations, for training as well as testing, we used randomly generated balanced datasets(equal number of positive and negative instances - 850 instances from each class) and repeated the process multiple number of times.

3.7.3 Effect of normalization schemes and kernels

It is known that large margin classifiers are sensitive towards feature scaling and standardization [**?**]. To study the same, we experiment with Attribute Normalization techniques and Kernel selection: Min-Max normalization and Z-Score normalization along with Linear and RBF Kernel.

[12]Pair of synsets not merged by annotators

We perform 5-fold validation i.e. we train the SVM on 5 randomly generated balanced datasets and test them again on a randomly balanced dataset disjoint from the training set. The table 3.8 documents the average results of the 5 runs. We report only FScore over both the classes as a measure of performance of the systems. For detailed results of the experiments, refer Appendix B.

Kernel	No Normalization	MinMaxNormalization	ZScoreNormalization
Linear	(0.31, 0.68)	(**0.73, 0.72**)	(0.05, 0.67)
RBF	(0.70, 0.37)	(**0.73**, 0.70)	(0.64, **0.72**)

Table 3.8: Studying Performance by varying Normalization Schemes and Kernels

For comparison purposes, we report the performance of the SVM systems, learnt for the 5-fold validation study (described above), on the Senseval-2 Dataset as the test set, in table 3.9.

Kernel	No Normalization	MinMaxNormalization	ZScoreNormalization
Linear	(0.15, **0.90**)	(**0.44**, 0.81)	(0.01, **0.90**)
RBF	(0.37, 0.46)	(0.40, 0.69)	(0.41, 0.85)

Table 3.9: SVM Performance on Senseval-2 Dataset

Observe that both feature normalization as well as kernel selection have a great influence on the classification performance. [**?**] report that if the data is not normalized the accuracy of an SVM can severely degrade. Therefore it is important to select the appropriate normalization and kernel for the task.

For most of the normalization-kernel combinations, the performance is biased towards a particular class. Among the normalization techniques, MinMax Normalization seems to give consistently good performance in both the kernels. So for further studies, we decided to use MinMax Normalization on features.

Linear Kernel gives better results on both the Senseval-2 dataset and the cross validation, and hence we choose linear kernel over RBF kernel.

3.7.4 Feature Analysis

We analyze our feature space in two ways. We evaluate Information Gain and Gain Ratio functions over the features and do a feature ablation study. The former tries to capture the discrimination ability of the feature by itself and the latter tries to measure how a feature corroborates with other features in the feature space.

3.7.4.1 Information Gain and Gain Ratio Study

Information Gain The Information Gain function is based on the notion of entropy, which characterizes the impurity of an arbitrary set of examples distributed among some classes. If an example is randomly selected from a set and its class c_i is announced, then the probability of this announcement is equal to $p_i = \frac{|c_i|}{|D|}$, and the amount of information it conveys is $-\log_2(p_i)$. The expected information provided by a announcement with respect to the class membership in a dataset D, having m classes with estimated class probabilities $p_1, \ldots, p_m$, is given by:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i) \tag{3.10}$$

The quantity $Info(D)$ measures the average amount of information(in bits) needed to identify the class of an example in D. We consider a similar measurement after D has been partitioned on attribute A in v parts, labeled as $D_1, \ldots D_v$. The amount of information needed to arrive at an exact classification after partitioning using that attribute is the weighted sum over subsets and is given by:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j) \tag{3.11}$$

The Information Gain is the expected reduction of information requirements caused by knowing the value of A and is given by:

$$Gain_A(D) = Info(D) - Info_A(D) \tag{3.12}$$

Gain Ratio The Information Gain function is biased towards tests with many outcomes. To counter the same, Gain Ratio modifies the Information Gain by taking into account the number and sizes of the partitions obtained in a test.

The *split information value* measures the potential information generated by dividing the dataset D into v bins, corresponding to v outcomes on attribute A.

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right) \tag{3.13}$$

The gain ratio is defined as:

$$GainRatio_A(D) = \frac{Gain_A(D)}{SplitInfo_A(D)} \tag{3.14}$$

Evaluation We computed all the features over the complete OntoNotes dataset without any normalization and evaluated the same using Information Gain and Gain Ratio as measures. Table 3.10 compares the value for all the features. We highlight the top 7 features according to both the attribute evaluators.

Feature	**Gain Ratio**	**Information Gain**
LCH	0.01288	**0.0323**
WUP	0.0148	**0.02899**
JCN	**0.0215**	0.02094
LIN	0.01943	0.02072
RES	0.01379	0.02335
AdapLesk	0.01688	**0.03456**
AdapLeskTani	**0.02306**	**0.03603**
AdapLeskTaniNoHypo	0.01685	**0.03014**
Common Lemma Count	0.00438	0.00394
SenseCount	0.00293	0.00293
SenseNum	0.0	0.0
lexFileSimilarity	0.01552	0.01143

mergeSP1_1	0.00282	0.00151
mergeSP1_2	**0.04195**	0.00103
mergeSP1_2_relaxed	**0.04709**	0.00119
mergeSP1_3	0.0	0.0
number of Common Hypernyms	**0.08833**	0.00965
autohyponymy	0.0	0.0
Domain-Cosine Similarity	**0.01997**	**0.04416**
Domain-l1 Distance	0.00445	0.00219
Domain-l2 Distance	0.00771	0.00238
OEDMerged	**0.0326**	**0.03123**
SentiWordNet-CosineSimilarity	0.0	0.0
SentiWordNet-l1 Distance	0.0	0.0
SentiWordNet-l2 Distance	0.0	0.0
CommonEnglishTranslations	0.00829	0.00671
CommonGermanTranslations	0.00732	0.00559
CommonSpanishTranslations	0.00547	0.00445
CommonItalianTranslations	0.00505	0.00418
CommonFrenchTranslations	0.00737	0.00634
CommonCatalanTranslations	0. 0.00657	0.00533
CommonDBpediaEntries	0.0	0.0

Table 3.10: Information Gain and Gain Ratio Based Evaluation

3.7.4.2 Feature Ablation Study

We divide our features in 6 categories: WordNet Similarity measures, WordNet based features, eXtended WordNet Domains features, BabelNet features, SentiWordNet features and Navigli OED Mappings.

We report the average F-Score of both the classes observed by removing that category of features from our feature space, retraining and retesting the classifiers

on randomly generated balanced datasets, keeping everything else the same. The SVMs are trained using linear kernel and features are normalized using MinMax Normalization for all the experiments reported in this study. The table 3.11 summarises the study.

Features Removed	**FScore Positive**	**FScore Negative**
WordNet Similarity Measures	0.6948	0.6784
WordNet Based Features	0.7227	0.7092
BabelNet Features	0.7232	0.7127
Domain Similarity Features	0.6814	0.6619
OED Feature	0.6957	0.7212
SentiWordNet Features	0.7262	0.7192
Without Removing Features	0.7262	0.7192

Table 3.11: Feature Ablation Study

3.7.4.3 Observations

WordNet Similarity Features : The similarity measures have a significant effect on the performance of the SVMs as can be observed from table 3.11. This highlights the importance of the underlying ontology structure of the WordNet which these similarity measures try to capture.

Note from table 3.10 that the gloss based features(the lesk variants - refer section 3.4.1.1) have high Information Gain values. This can be attributed to the fact that the annotators primarily rely on gloss descriptions to interpret synsets. In the relatedness study by [**?**], the annotators only had glosses as evidence to decide if senses were "related" or not.

WordNet Based Features : Among the WordNet based features, the features relating the synsets to their hypernyms like the "SP1_2 merge heuristics", the number of common hypernyms etc. seem to be discriminatory. This is understandable as the hypernym related feaures capture the notion of semantic generalization, which

is essential to undestand a sense.

BabelNet Features : The objective of using multilingual features was to test whether the translation equivalences are powerful enough to capture semantic associations between two word senses. Low values of the Information Gain and Gain Ratio of the BabelNet features reflect that the above heuristic is a weak indicator for sense-merging.

Using mapping to DBPedia entries as a feature was an effort to harness the DBpedia Knowledge Base [13]. But we observe from table 3.10 that the feature is not that useful. A better use of the DBpedia ontology would be to estimate the similarity of mapped concepts using the underlying hierarchy.

Domain Similarity Features : Intuitively, as an annotator, approximately matching the domain of two senses serves as a strong cue about whether the two senses are semantically related enough to be merged. This is justified by the high info-gain and gain-ratio values for the domain similarity features.

Oxford English Dictionary Mapping : [**?**] have already shown the effectiveness of this mapping in a WSD task based setting. High values of Information Gain and Gain Ratio support the same. The problem we face with the feature is its incompleteness as not all the word senses are clustered by mapping to Oxford English Dictionary.

SentiWordNet Features : Another interesting set of features is the SentiWordNet based features. Their removal doesn't affect the system's performance which can be attibuted to the fact that most of the noun synsets in the SentiWordNet project are described as objective concepts. Their non-discriminatory nature is substantiated by the Information Gain and Gain Ratio based study as well (refer table 3.10).

[13] `http://dbpedia.org/About`

3.8 Discussion

In this section, we would like to address some concerns regarding the similarity function learnt.

3.8.1 Inconsistent Predictions

Using the outputs of the SVM learnt directly as the similarity distance poses the problem of inconsistent predictions i.e. it can happen that for three synsets A, B and C, $sim_{SVM}(A, B) > 0$ and $sim_{SVM}(B, C) > 0$ while $sim_{SVM}(A, C) < 0$. Such inconsistencies, though rare, can happen. Even the human annotators involved in preparation of Senseval-2 dataset [**?**] have made such errors. This motivates us to utilize the WordNet structure to correct such inconsistencies.

3.8.2 Coverage of the SVM

The training data for the SVM is not representative of the WordNet synsets because we trained only on the synset pairs that have atleast one lemma in common. It is interesting to note here that the number of synsets which contain atleast one polysemous lemma is only 33155 out of total 82115 synsets. This questions the idea of using the SVM models learnt as generic synset similarity estimators.

[**?**] addresses this issue by taking similarity between synsets not sharing any word as 0 and for the synsets sharing atleast a word as the prediction by the SVM, for the purpose of sense-merging. Because of the heterogeneous nature of the similarity defined by [**?**], it does not serve as a generic synset similarity measure.

3.8.3 Insufficient Data for Learning

The SVMs were learnt on randomly selected balanced datasets of 1700 instances with 850 instances of each class. Though the results are promising, the number of training instances is small as compared to the total number of synsets involved (33155), which makes it tough to judge whether the similarity metric learnt is generic

enough or not.

3.9 Conclusions and Future Work

[**?**] performed an annotation study in which 3 native english speakers were asked to label around 300 sense pairs as "related", "unrelated" or "don't know". The inter-annotator F-Scores were (0.7926, 0.5454, 0.4874)[14] [**?**] [**?**]. These figures highlight that humans differ in their tendency to split or lump senses and that the task is inherently a difficult one.

Significant advancement of supervised learning algorithms over the last two decades and their ability to capture the relative importance of features essential to the task in hand inspired us to utilize their potential in understanding the importance of the various features in merging synsets.

The use of external corpora for supervision is motivated by the fact that we are not able to fully capture the information in WordNet for e.g. we are not able to utilize the gloss of the synset beyond lexical measures. By enriching the semantic information of the synsets using features like belongingness to different domains, sentiment associated with them etc. we are able to improve the performance of our systems.

The evaluation suggests that there is a need to enrich WordNet along with the production of additional resources to better understand word senses. Some such efforts include augmenting WordNet with teleological links [15], morphological and semantic information [**?**].

[14]Since the annotation was done by native speakers and not experienced linguists or lexicographers, we can expect a slightly higher inter-annotator F-Score for the task.

[15]`http://wordnetcode.princeton.edu/standoff-files/teleological-links.xls`

Chapter 4

Semi-Supervised Synset Similarity

In this chapter, we describe the idea of SimRank, its personalization and application to learning synset similarity in a semi-supervised fashion. Further we discuss how to coarsen the WordNet taxonomy and evaluate the clustering obtained in a WSD task based setting.

4.1 Motivation

The small coverage of the Support Vector Machines that we saw in section 3.2 makes them unsuitable to be used as a generic synset similarity estimator. To learn the similarity between synsets which do not share a word, we wish to utilize the relations encoded in WordNet as well as the information learnt using supervision. The ability of the Personalized SimRank framework to propogate the seeded information using semantic links between concepts allows us to learn complete synset similarity estimates.

4.2 SimRank

4.2.1 Introduction

SimRank [**?**] is a graph based similarity measure applicable in any domain with object-to-object relationships, with intuition that "**two objects are similar if**

they are referenced by similar objects" [**?**] [**?**]. Since SimRank has a recursive intuition, the base cases play an important role here.

Before discussing the model, let us introduce some notation for convenience. For a graph $G(V, E)$, for each node $v \in V$, we define the following:

- $I(v)$ is a set consisting of in-neighbours of node v i.e.

$$I(v) = \{w \in V | (w, v) \in E\} \tag{4.1}$$

- $O(v)$ is a set consisting of out-neighbours of node v i.e.

$$O(v) = \{u \in V | (v, u) \in E\} \tag{4.2}$$

Individual members of $O(v)$ and $I(v)$ are referred to as $O_i(v)$, $1 \leq i \leq |O(v)|$ and $I_j(v)$, $1 \leq j \leq |I(v)|$.

Let $s(\alpha, \beta)$ denote the similarity score between objects α and β. The SimRank equation is given by:

$$s(\alpha, \beta) = \begin{cases} 1 & \text{if } \alpha = \beta \\ 0 & \text{if } I(\alpha) = \phi \text{ or } I(\beta) = \phi \\ \frac{C}{|I(\alpha)||I(\beta)|} \sum_{i=1}^{|I(\alpha)|} \sum_{j=1}^{|I(\beta)|} s(I_i(\alpha), I_j(\beta)) & \text{otherwise} \end{cases} \tag{4.3}$$

Here C is a constant in range $(0, 1)$ and is called the decay factor.

4.2.2 Solution and its Properties

The solution to the SimRank equation 4.3 for a graph $G(V, E)$ is reached by iteration to a fixed-point. For each iteration k, we keep $|V|^2$ entries $S_k(*, *)$, where $S_k(\alpha, \beta)$ is the estimate of similarity between α and β on k^{th} iteration. We start with $S_0(*, *)$ which is 1 for singleton nodes like (x, x), 0 otherwise.

We successively compute $S_{k+1}(*,*)$ based on $S_k(*,*)$:

$$S_{k+1}(\alpha,\beta) = \begin{cases} 1 & \text{if } \alpha = \beta \\ 0 & \text{if } I(\alpha) = \phi \text{ or } I(\beta) = \phi \\ \frac{C}{|I(\alpha)||I(\beta)|} \sum_{i=1}^{|I(\alpha)|} \sum_{j=1}^{|I(\beta)|} S_k(I_i(\alpha), I_j(\beta)) & \text{otherwise} \end{cases} \tag{4.4}$$

Let us highlight some properties of SimRank covered in [**?**] and [**?**]:

1. A solution $s(*,*)$ to SimRank equations always exist and is unique, and $s(*,*) \in [0,1]$

2. For each k, $S_k(*,*)$ is upper bounded by the SimRank function $s(*,*)$ i.e.

$$S_k(\alpha,\beta) \leq s(\alpha,\beta) \tag{4.5}$$

3. Iterative functions $S_k(*,*)$ converge to SimRank function $s(*,*)$ i.e.

$$\lim_{k\to\infty} S_k(\alpha,\beta) = s(\alpha,\beta) \tag{4.6}$$

4. The difference between the SimRank scores and iterative similarity scores decreases exponentially in the number of iterations and uniformly for every pair of nodes i.e.

$$s(\alpha,\beta) - S_k(\alpha,\beta) \leq C^{k+1} \quad \forall \alpha,\beta \in V;\ k = 0,1,2\ldots \tag{4.7}$$

4.2.3 Random Surfer Pair Model

[**?**] show that the SimRank similarity score $s(\alpha,\beta)$ measures how soon two random surfers are expected to meet at the same node if they start at nodes α and β and randomly walk the graph backwards.

$$s(\alpha,\beta) = \sum_{t:(\alpha,\beta)\rightsquigarrow(x,x)} P[t]C^{l(t)} \tag{4.8}$$

More formally, $s(\alpha, \beta)$ is the *expected f-meeting distance* traveling back edges, between the nodes α and β with $f(z) = C^z$.

4.3 Personalized Weighted SimRank

4.3.1 Weighted SimRank

Many times while working with real data, we observe relationships of different types between objects, which are likely to have different weights associated with them. For learning similarity between objects having such underlying structure, we discuss a weighted version of the SimRank method proposed in [**?**] and discussed in section 4.2. For convenience, let us define $W_I(n)$ and $W_O(n)$, for a node n as follows:

$$W_O(n) = \sum_{i=1}^{|O(n)|} w(n, O_i(n)) \tag{4.9}$$

$$W_I(n) = \sum_{i=1}^{|I(n)|} w(I_i(n), n) \tag{4.10}$$

The recursive equation for weighted SimRank is as follows:

$$s(\alpha, \beta) = \frac{C}{W_I(\alpha) W_I(\beta)} \sum_{i=1}^{|I(\alpha)|} \sum_{j=1}^{|I(\beta)|} w(I_i(\alpha), \alpha) \cdot w(I_j(\beta), \beta) \cdot s(I_i(\alpha), I_j(\beta)) \tag{4.11}$$

The relation 4.11 is obtained by extending the Random Surfer Pairs Model of SimRank and is discussed in more detail in Appendix C.

4.3.2 Personalizing SimRank

In many scenarios, while working with objects, we don't have complete information about the objects and thus have similarities for only some pair of objects. These similarities may be independently learnt and may not directly conform with the underlying graph. In such situations, we would like to get a more complete and consistent similarity metric between objects but we would like to use the information

given to us as well. For the same, we propose a personalized framework for SimRank, in which we bias the SimRank by changing the initialization. If we know similarities of some pairs, we fix them in our set of equations and let the rest of the values be automatically learned by the system.

Lets call the map of node pairs to their similarity values as $InitStore$. All the singleton nodes like (x, x) have value equal to 1. The system of equations is defined as follows:

$$s(\alpha,\beta) = \begin{cases} 1 & \text{if } \alpha = \beta \\ InitStore[(\alpha,\beta)] & \text{if } (\alpha,\beta) \in InitStore \\ \frac{C}{W_I(\alpha)W_I(\beta)} \sum_{i=1}^{|I(\alpha)|} \sum_{j=1}^{|I(\beta)|} w(I_i(\alpha),\alpha) \cdot w(I_j(\beta),\beta) \cdot s(I_i(\alpha), I_j(\beta)) & \text{otherwise} \end{cases} \tag{4.12}$$

In the personalized framework, we have no constraints over the initialization as long as all values initialized are in range $[0, C]$.

4.3.3 Solution of Personalized SimRank

A solution to the equations 4.12 for a graph $G(V, E)$ is computed on lines of the method to solve equation 4.3 (refer section 4.2.2). The successive computation of $S_{k+1}(*,*)$ from $S_k(*,*)$ is calculated as follows:

$$S_{k+1}(\alpha,\beta) = \begin{cases} 1 & \text{if } \alpha = \beta \\ InitStore[(\alpha,\beta)] & \text{if } (\alpha,\beta) \in InitStore \\ \frac{C}{W_I(\alpha)W_I(\beta)} \sum_{i=1}^{|I(\alpha)|} \sum_{j=1}^{|I(\beta)|} w(I_i(\alpha),\alpha) \cdot w(I_j(\beta),\beta) \cdot S_k(I_i(\alpha), I_j(\beta)) & \text{otherwise} \end{cases} \tag{4.13}$$

All the properties of solution to SimRank equation are applicable to Personalized SimRank solution as well (refer section 4.2.2). We prove all these properties for Personalized SimRank in Appendix D.

4.4 Personalized SimRank for Learning Synset Similarity

4.4.1 Algorithm Outline

The Personalized SimRank framework requires an underlying graph $G(V, E)$, where V is the set of objects to be clustered and E is the set of semantic links connecting these objects and an $InitStore$ which contains the similarity values over some pairs from $V \times V$ learnt or known beforehand. Note that the values in the $InitStore$ have an upper bound of C.

For learning synset similarity, V is the set of synsets to be clustered and E is the set of WordNet relations connecting these synsets. We use the *Hypernymy*, *Hyponymy*, *Meronymy* and *Holonymy* relations of WordNet as the semantic links. We seed the $InitStore$ as follows:

- We predict the similarity values of all the synset pairs which share at least one word using the Support Vector Machine learnt from synset-merging data from OntoNotes [**?**] as described in section 3.5.

- We transform the SVM predictions to the range $[0, 1]$ using the sigmoid model learnt on the OntoNotes dataset. (see section 4.4.2)

- We scale the posterior probabilities obtained to range between $[0, C]$ by linear scaling, where C is the SimRank decay parameter.

4.4.2 Estimating Posterior Probabilities from SVM Scores

Given a set of training examples $x_i \in \mathbb{R}^n, i = 1, \ldots, t$, labeled by $y_i \in \{-1, +1\}$ with N_+ positive examples and N_- negative examples, a Support Vector Machine finds a maximum margin decision boundary $f(x)$ whose sign serves as a label prediction for any test example x. Instead of label prediction, many systems, as in our case: SimRank Initialization, require posterior probability $Pr(y = +1|x)$ estimate. [**?**]

proposed approximating the posterior by a sigmoid function

$$Pr(y = +1|x) \approx P_{A,B}(f(x)) \equiv \frac{1}{1 + exp(Af(x) + B)} \tag{4.14}$$

To avoid overfitting of the sigmoid, an out-of-sample model is used i.e. for each training example target values t_is are used (instead of 1 and 0), where

$$t_i = \begin{cases} \frac{N_+ + 1}{N_+ + 2} & \text{if } y_i = +1 \\ \frac{1}{N_- + 2} & \text{if } y_i = -1 \end{cases} \tag{4.15}$$

Let each f_i be shorthand for $f(x_i)$. The best parameter setting $z^* = (A^*, B^*)$ is determined by solving the following regularized maximum likelihood problem:

$$\min_{z=(A,B)} F(z) = -\sum_{i=1}^{t} (t_i \log(P_{A,B}(f_i)) + (1 - t_i) \log(1 - P_{A,B}(f_i))) \tag{4.16}$$

[**?**] indicated that for solving equation 4.16, any method for unconstrained optimization can be used. We use Newton's method with backtracking, as proposed in [**?**], as it avoids numerical difficulties faced by [**?**]. The pseudocode for the implementation is available in Appendix E.

4.4.3 Importance of Parameter C

The parameter C in Personalized SimRank affects the algorithm in many ways:

- **Maximum Similarity between distinct objects**: According to the SimRank equations, the maximum similarity between two distinct objects can be at most C.

$$s(a,b) = \frac{C}{W_I(a)W_I(b)} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} w(I_i(a),a) \cdot w(I_j(b),b) \cdot \underbrace{s(I_i(a), I_j(b))}_{\leq 1} \quad (4.17)$$

$$\leq \frac{C}{W_I(a)W_I(b)} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} w(I_i(a),a) \cdot w(I_j(b),b) \quad (4.18)$$

$$\leq \frac{C}{W_I(a)W_I(b)} \left(\sum_{i=1}^{|I(a)|} w(I_i(a),a) \right) \cdot \left(\sum_{j=1}^{|I(b)|} w(I_j(b),b) \right) \quad (4.19)$$

$$\leq C \quad (4.20)$$

- **Rate of convergence**: The rate of convergence of the iterative method to find the solution for SimRank equations is dependent on C as there is exponential decrease in the difference between SimRank theoretical and iterative similarity scores in the number of iterations. In short, the higher the value of C, the slower is the convergence.

To understand the importance of the parameter C, we vary the value of C in our experiments and empirically study the results obtained.

4.5 Coarsening WordNet

In this section we would discuss our method to cluster synsets of WordNet, which would give us coarser senses for words. The task in hand can be described as follows: Assuming we have a similarity metric which gives us similarity between each pair of synsets, we need to cluster the synsets where the granularity of the clustering can be adjusted according to the needs of the application.

We use the similarity metric learnt using the personalized SimRank model as described in section 4.4. We construct a graph $G(V, E)$ where the vertices V are the synsets of the WordNet graph and edges E are obtained by thresholding the similarity metric. On this graph, we find connected components, which gives us a partition over synsets. The algorithm 1 summarises our method:

Algorithm 1 Coarsening WordNet

Input: WordNet Graph $WN(V, E)$, Similarity Metric Sim and threshold t
Output: Partition over V
$E' \leftarrow \emptyset$
for all Synset $s_i \in V$ **do**
    for all Synset $s_j \in V$ **do**
        if $Sim(s_i, s_j) \geq t$ **then** $E' \leftarrow E' \cup (s_i, s_j)$
        end if
    end for
end for
$C \leftarrow ConnectedComponents(G(V, E'))$
return C

We use the clustering obtained to derive coarse senses of words. For a word, all its senses in the same cluster are merged and act as a coarse sense. Algorithm 2 summarises the querying method:

Algorithm 2 Querying Senses of a word

Input: WordNet Graph $WN(V, E)$, Word w and Partition P over V
Output: Set of coarse senses C
$L \leftarrow ListSenses_{WN}(w)$
$C \leftarrow \emptyset$
for all Synset $s_i \in L$ **do**
    $C \leftarrow C \cup \{s_i\}$
end for
for all Synset $s_i \in L$ **do**
    for all Synset $s_j \in L$ **do**
        if $s_i \neq s_j$ and $P(s_i, s_j) == 1$ **then** $C \leftarrow Merge(C, s_i, s_j)$
        end if
    end for
end for
return C

4.6 Experimental Setup and Evaluation

4.6.1 Estimating Posterior Probabilities from SVM Scores

We learn the parameters A and B of the sigmoid transforming SVM predictions to posterior probabilities (see section 4.4.2). Using the same data set that was used to train the model we want to calibrate will introduce unwanted bias. So we used an independent calibration set in order to get good posterior probabilities [**?**].

We address this problem by calibrating on an independently generated random balanced subsets from OntoNotes (see section 3.3.2). The SVM predictions are obtained using the SVMs trained over random subsets of the OntoNotes data (see section 3.7). Since all the SVMs were trained on randomly generated datasets, we selected the SVM that performed the best in cross validation.

The values of A and B obtained are -1.1655 and 0.0222 respectively. Using these values, the SVM prediction of value 0 gets mapped to 0.4944.

4.6.2 Semi-Supervised Similarity Learning

We learn similarity models using the SimRank variant described in section 4.4.1. [**?**] use $C = 0.8$ and report that 5-6 iterations are enough. [**?**] suggest to use lower values of C or more number of iterations. We vary the values taken by C as 0.6, 0.7 and 0.8. We run all the systems for 10 iterations to avoid convergence issues.

We use the *Hypernymy*, *Hyponymy*, *Meronymy* and *Holonymy* relations of WordNet as the semantic links. All the semantic links used in all the experiments have uniform weight unity. For implementation, we used EZGraph Java library[1].

4.6.3 Coarsening WordNet

We assess the effect of automatic synset clustering on the English all-words task at Senseval-3 [**?**] (this evaluation is similar to the evaluation used by [**?**] and [**?**]). The task asked WSD systems to select the apt sense for 2,041 content words in running texts comprising of 351 sentences. Since we focus on nouns, we used the 890 instances labelled with noun as the part of speech.

We consider the three best performing WSD systems: GAMBL [**?**], SenseLearner [**?**] and Koc University [**?**] - and the best unsupervised system: IRST-DDD [**?**] submitted in the task. The answer by the system is given full credit if it belongs to the cluster of the correct answer.

Observe that any clustering will only improve the WSD performance. Therefore

[1]ezgraph - Easy to Use Graph Analysis Library: `https://code.google.com/p/ezgraph/`

to assess the improvement obtained because of our particular clustering, we calculate the expected performance for a random clustering at the same granularity as our clustering and study the improvement over random clustering instead.

Score for a random clustering is calculated as follows: Let the word to be disambiguated have N senses, each mapped to a unique synset. Let the clustering of these N synsets on a particular granularity give us k clusters $C_1, \ldots C_k$. The expectation that an incorrectly chosen sense and the actual correct sense would be in cluster C_i in this clustering is $\frac{|C_i|(|C_i|-1)}{N(N-1)}$. Using linearity of expectation, we can say that the chances of the two synsets belonging to same cluster would be

$$\frac{\sum_{i=1}^{k} |C_i|(|C_i|-1)}{N(N-1)} \tag{4.21}$$

We experiment with $C = 0.6$, 0.7 and 0.8. The SVM probability boundaries when scaled to $[0, C]$ for these values are 0.30, 0.35 and 0.40. So for finding the threshold giving the best improvement against random clustering baseline, we use the search space $[C - 0.35, C]$. The performance of the systems at these thresholds for different values of C is reported in table 4.1.

C	**System**	**F-Score**	**Threshold**	**CCC**	**Random**	**Improvement**
0.6	GAMBL	0.7116	0.36	0.9031	0.8424	0.0607
	SenseLearner	0.7104	0.37	0.8824	0.8305	0.0518
	KOC University	0.7191	0.37	0.8924	0.8314	0.0610
	IRST-DDD	0.6367	0.35	0.8731	0.8013	0.0718
0.7	GAMBL	0.7116	0.52	0.8453	0.7864	0.0589
	SenseLearner	0.7104	0.49	0.8541	0.8097	0.0444
	KOC University	0.7191	0.52	0.8448	0.7911	0.0538
	IRST-DDD	0.6367	0.49	0.7970	0.7402	0.0568
0.8	GAMBL	0.7116	0.59	0.8419	0.7843	0.0577
	SenseLearner	0.7104	0.56	0.8439	0.7984	0.0455
	KOC University	0.7191	0.59	0.8414	0.7879	0.0535

	IRST-DDD	0.6367	0.47	0.8881	0.8324	0.0557

Table 4.1: Improvement in Senseval-3 WSD performance using Connected Component Clustering Vs Random Clustering at the same granularity

Commenting theoretically about the impact of C on the performance is tough as by changing C we are changing all the $|V|^2$ simultaneous equations to be solved. Empirically we observe that improvements over baseline keep on decreasing on increasing C across all the systems. It might be because of the slow convergence of SimRank for higher values of C.

We observe from the figures 4.1 and 4.2 that by varying thresholds, the improvement of the Connected Components Clustering over the random clustering baseline at the same granularity first increases and then decreases.

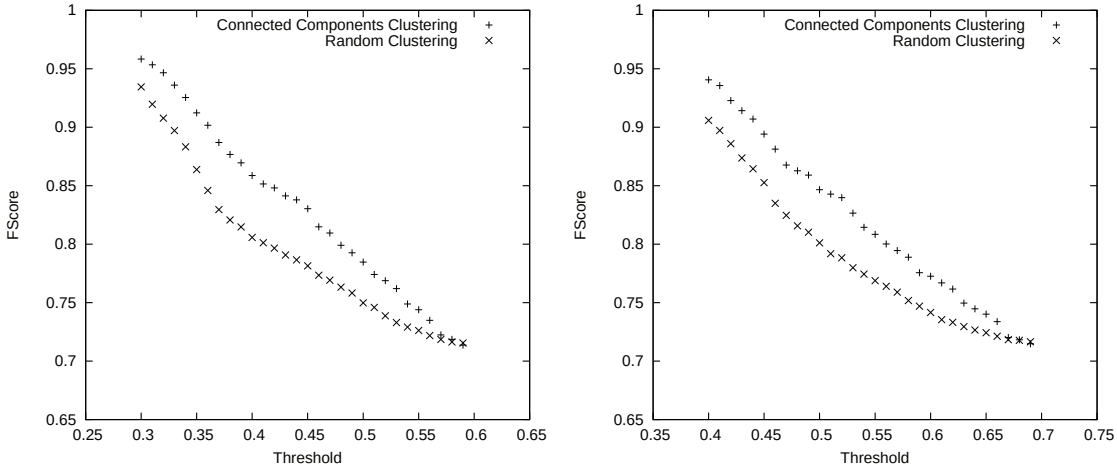
Across supervised and unsupervised systems, we observe that the unsupervised systems obtain higher improvements than the supervised systems which can be attributed to fact that the unsupervised system used was underperforming compared to the supervised systems in the fine grained WSD task setting.

4.7 Conclusions and Future Work

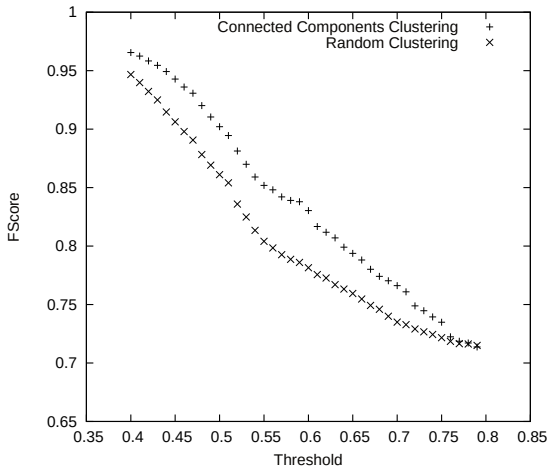
4.7.1 Conclusions

The method discussed in the chapter learns a model for calculating synset similarity utilizing the taxonomy information and information learnt from manually obtained sense clustering. The framework obtained is generic and can be applied to other parts of speech as well.

For coarsening senses, we proposed one of the simplest approaches to cluster senses but it can be made to work with any clustering algorithm to generate coarse senses. We show that the clustering obtained by partitioning synsets in connected components gives us a maximum improvement of 5.78% on supervised systems and 7.18% on unsupervised system.

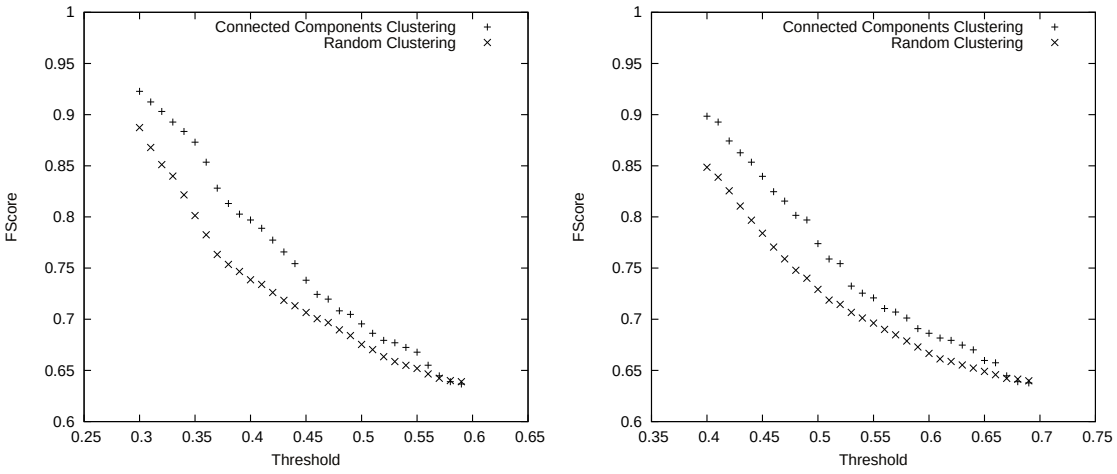


(a) (b)

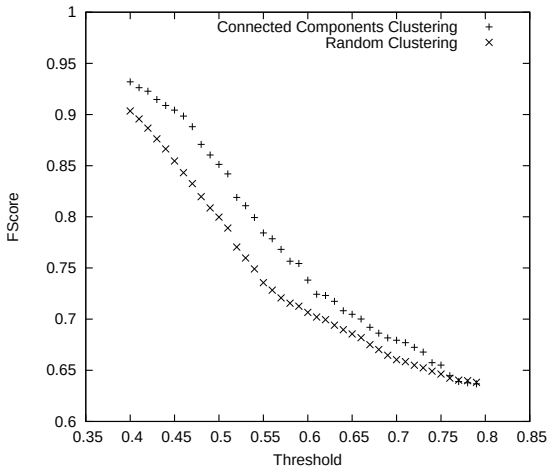


(c)

Figure 4.1: Improvement in average performance of best 3 Supervised Systems in Senseval-3 using Connected Component Clustering Vs Random Clustering at the same granularity with C = (a) 0.6, (b) 0.7 and (c) 0.8



(a) (b)



(c)

Figure 4.2: Improvement in performance of best Unupervised System in Senseval-3 using Connected Component Clustering Vs Random Clustering at the same granularity with C = (a) 0.6, (b) 0.7 and (c) 0.8

4.7.2 Future Work

4.7.2.1 Differentiating WordNet relations

We use the WordNet relations *Hypernymy*, *Hyponymy*, *Meronymy* and *Holonymy* without any differentiation i.e. weights of all the links are unity. If we can grade the weights of the relations based on their relative importances, we can expect an improvement in the system. One way can be to perform cognitive experiments and obtain the weights of relations from the feedback of the annotators. An alternative way can be learn weights in a task based setting using an approach similar to [**?**].

4.7.2.2 Speeding up the implementation

We used the naive implementation of SimRank, proposed in [**?**], for implementing Personalized SimRank. Faster approximate scalable implementations of SimRank like [**?**], [**?**] etc. can also be adapted appropriately for personalization and thus speed up the algorithm.

Chapter 5

Conclusions and Future Work

We presented a classifier for automatic synset merging using a rich collection of features derived from the information encoded in WordNet and other lexical resources mapped to WordNet available in the public domain. We show the effectiveness of the features used in classification and observe that the features relating the synset to the domain it belongs, obtained from eXtended WordNet Domains project [**?**] provide a useful cue for sense merging. Since the classifier was learnt only on synset pairs sharing at least one word, we sought out to estimate synset similarity for other pairs.

To learn similarity between synset pairs not sharing a word, we proposed a variant of SimRank [**?**]: Personalized Weighted SimRank. With this model, we coarsened senses by partitioning the synsets using Connected Components. We study the clusters obtained by varying the decay parameter and report a maximum performance improvement of 5.78% on supervised systems and 7.18% on an unsupervised system.

The synset similarity measure proposed can be extended for other POS as well. The generic nature of the similarity gives us the flexibility of using other clustering algorithms for experimentation.

5.1 Future Work

5.1.1 Correcting Inconsistencies

We discussed in section 3.8.1 the problem of inconsistent judgements by a classifier, Support Vector Machines in our case. Since this issue can arise with any classifier, we need to focus on correcting such mistakes either automatically or semi-automatically. An automatic attempt can make use of the WordNet ontology to decide what to do in case of such inconsistencies.

A semi-supervised approach to correct inconsistencies can be used in which we ask annotators to label such pairs. Appending these annotations obtained to the training dataset, we relearn our classifiers hoping that now they will be able to learn decision boundaries in a better manner. The idea is closely related to Active Learning frameworks.

The inconsistencies also question whether the data from which we learn the model is sufficient or not. The dearth of manually sense tagged resources, be it sense annotated texts or binary/graded sense similarity judgements, is already acknowledged by the NLP community. This is a major barrier for a knowledge-intensive task like WSD and is referred as the *knowledge acquisition bottleneck*.

5.1.2 Graph based similarity estimation

We proposed a simple variant of SimRank [**?**] to estimate similarity between synsets. We show that the clustering obtained using the similarity metric obtained indeed performs well. This encourages us to study more about Graph based similarity learning methods as they enable us to employ available wide-coverage knowledge bases.

5.1.3 Graded Evaluation of WSD Task

Word Sense Disambiguation is the task of selecting the apt sense for a word among its listed senses. The evaluation frameworks used in literature use a binary scoring

i.e. if the sense marked as answer by the system matches the sense annotated by humans, then it receives full credit otherwise it receives no credit. The problem with this scoring is that it penalizes all the wrong answers equally.

Consider the noun senses of the word *fish*(refer example 1). We think that a WSD system mistagging the astrology sense of the word *fish* with its food sense should be penalized more than a system confusing between its two astrology related senses. So, we propose that the scoring should be graded instead of binary. Let L be the list of senses of the word, CS be the correct sense and Sim be a similarity estimator between senses. For eg. the score given to the assigned sense $AS \in L$ can be

$$score(AS, CS, L) = Sim(AS, CS) \tag{5.1}$$

Appendices

Appendix A

Lexicographer Files

The names of the lexicographer files and their corresponding file numbers are listed below along with a brief description each file's contents. The table is reproduced from the WordNet manual [1].

File Number	Name	Contents
00	adj.all	all adjective clusters
01	adj.pert	relational adjectives (pertainyms)
02	adv.all	all adverbs
03	noun.Tops	unique beginner for nouns
04	noun.act	nouns denoting acts or actions
05	noun.animal	nouns denoting animals
06	noun.artifact	nouns denoting man-made objects
07	noun.attribute	nouns denoting attributes of people and objects
08	noun.body	nouns denoting body parts
09	noun.cognition	nouns denoting cognitive processes and contents
10	noun.communication	nouns denoting communicative processes and contents

[1] `http://wordnet.princeton.edu/man/lexnames.5WN.html`

11	noun.event	nouns denoting natural events
12	noun.feeling	nouns denoting feelings and emotions
13	noun.food	nouns denoting foods and drinks
14	noun.group	nouns denoting groupings of people or objects
15	noun.location	nouns denoting spatial position
16	noun.motive	nouns denoting goals
17	noun.object	nouns denoting natural objects (not man-made)
18	noun.person	nouns denoting people
19	noun.phenomenon	nouns denoting natural phenomena
20	noun.plant	nouns denoting plants
21	noun.possession	nouns denoting possession and transfer of possession
22	noun.process	nouns denoting natural processes
23	noun.quantity	nouns denoting quantities and units of measure
24	noun.relation	nouns denoting relations between people or things or ideas
25	noun.shape nouns	denoting two and three dimensional shapes
26	noun.state nouns	denoting stable states of affairs
27	noun.substance	nouns denoting substances
28	noun.time	nouns denoting time and temporal relations
29	verb.body	verbs of grooming, dressing and bodily care

30	verb.change	verbs of size, temperature change, intensifying, etc.
31	verb.cognition	verbs of thinking, judging, analyzing, doubting
32	verb.communication	verbs of telling, asking, ordering, singing
33	verb.competition	verbs of fighting, athletic activities
34	verb.consumption	verbs of eating and drinking
35	verb.contact	verbs of touching, hitting, tying, digging
36	verb.creation	verbs of sewing, baking, painting, performing
37	verb.emotion	verbs of feeling
38	verb.motion	verbs of walking, flying, swimming
39	verb.perception	verbs of seeing, hearing, feeling
40	verb.possession	verbs of buying, selling, owning
41	verb.social	verbs of political and social activities and events
42	verb.stative	verbs of being, having, spatial relations
43	verb.weather	verbs of raining, snowing, thawing, thundering
44	adj.ppl	participial adjectives

Table A.1: Lexicographer Files and Description

Appendix B

Results for SVM Experiments

The appendix jots down the performance of the SVMs learnt on random datasets as described in section 3.7.3. The performance criteria are Precision, Recall, FScore (on both Positive and Negative classes) and Accuracy.

Performance Measure	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5
Accuracy	0.5687	0.5769	0.5646	0.5755	0.5467
PrecisionP	0.7404	0.75	0.7117	0.7570	0.8542
RecallP	0.2115	0.2308	0.2170	0.2225	0.1126
FScoreP	0.3291	0.3529	0.3326	0.3439	0.1990
PrecisionN	0.5401	0.5454	0.5381	0.5443	0.525
RecallN	0.9258	0.9231	0.9121	0.9286	0.9808
FScoreN	0.6822	0.6857	0.6769	0.6863	0.6839

Table B.1: Linear Kernel - No Normalization

Performance Measure	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5
Accuracy	0.7019	0.7376	0.7253	0.7294	0.7198
PrecisionP	0.6889	0.7357	0.7265	0.7158	0.7222
RecallP	0.7363	0.7418	0.7225	0.7610	0.7143
FScoreP	0.7118	0.7387	0.7245	0.7377	0.7182
PrecisionN	0.7168	0.7396	0.7240	0.7449	0.7174

RecallN	0.6676	0.7335	0.7280	0.6978	0.7253
FScoreN	0.6913	0.7366	0.7260	0.7206	0.7213

Table B.2: Linear Kernel - MinMax Normalization

Performance Measure	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5
Accuracy	0.5096	0.5151	0.5027	0.5069	0.5192
PrecisionP	0.7333	0.9231	0.75	1.0	0.9375
RecallP	0.03022	0.0330	0.0082	0.0137	0.0412
FScoreP	0.05805	0.0637	0.0163	0.0271	0.0789
PrecisionN	0.5049	0.5077	0.5014	0.5035	0.5098
RecallN	0.9890	0.9972	0.9972	1.0	0.9972
FScoreN	0.6685	0.6728	0.6673	0.6697	0.6747

Table B.3: Linear Kernel - ZScore Normalization

Performance Measure	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5
Accuracy	0.6003	0.6058	0.5797	0.5948	0.5934
PrecisionP	0.5588	0.5626	0.5459	0.5554	0.5556
RecallP	0.9533	0.9505	0.9478	0.9505	0.9341
FScoreP	0.7046	0.7068	0.6928	0.7011	0.6967
PrecisionN	0.8411	0.8407	0.8021	0.8286	0.7931
RecallN	0.2472	0.2610	0.2115	0.2390	0.2527
FScoreN	0.3822	0.3983	0.3348	0.3710	0.3833

Table B.4: RBF Kernel - No Normalization

Performance Measure	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5
Accuracy	0.7212	0.7198	0.7184	0.7308	0.7033
PrecisionP	0.7007	0.7083	0.6897	0.6972	0.6787
RecallP	0.7720	0.7472	0.7939	0.8159	0.7720
FScoreP	0.7346	0.7273	0.7382	0.7519	0.7224

PrecisionN	0.7462	0.7326	0.7573	0.7781	0.7357
RecallN	0.6703	0.6923	0.6428	0.6456	0.6346
FScoreN	0.7062	0.7119	0.6954	0.7057	0.6814

Table B.5: RBF Kernel - MinMax Normalization

Performance Measure	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Dataset 5
Accuracy	0.6731	0.6992	0.6648	0.6882	0.6799
PrecisionP	0.7316	0.7757	0.7325	0.7354	0.7347
RecallP	0.5467	0.5604	0.5192	0.5879	0.5632
FScoreP	0.6258	0.6507	0.6077	0.6534	0.6376
PrecisionN	0.6382	0.6559	0.6276	0.6568	0.6459
RecallN	0.7994	0.8379	0.8104	0.7885	0.7967
FScoreN	0.7097	0.7358	0.7074	0.7166	0.7134

Table B.6: RBF Kernel - ZScore Normalization

Appendix C

Weighted SimRank

We discuss the weighted SimRank Model here along with the theoretical interpretation using Random Surfer-Pairs Model.

C.1 Random Surfer-Pairs Model

We extend the Random Surfer-Pairs Model and Expected-f Meeting Distance discussed in [**?**] to a weighted graph. We define $s'(\alpha, \beta)$, the similarity between α and β in G based on *expected-f meeting distance*, with $f(z) = c^z$, as

$$s'(a, b) = \sum_{t:(a,b)\rightsquigarrow(x,x)} P[t]c^{l(t)} \tag{C.1}$$

where c is the a positive constant less than 1. If there is no tour from (α, β) to any singleton node, the summation is taken to be 0. Note from equation C.1 that $s'(\alpha, \beta) \in [0, 1]$ for all α, β, and that $s'(\alpha, \beta) = 1$ if $\alpha = \beta$.

C.2 Equivalence

For presentation ease, let us assume that all edges in the graph G have been reversed. So following an edge in the reversed graph is same as moving one step backwards in the original graph.

We compute the expected-f meeting distance on lines of computation of expected distance in the graph. Suppose a surfer is at $u \in V$. In the next step, he chooses one of $O_1(u), \ldots, O_{|O(u)|}(u)$, say $O_i(u)$, proportional with the corresponding edge weight i.e. with probability $\frac{w(u,O_i(u))}{W_O(u)}$. On choosing $O_i(u)$, the process is repeated.

We'll now show that $s'(\alpha, \beta) = s(\alpha, \beta)$

$$
\begin{aligned}
s'(\alpha,\beta) &= \sum_{z=1}^{|O((\alpha,\beta))|} \sum_{t':O_z((\alpha,\beta))\rightsquigarrow(x,x)} P[(\alpha,\beta) \to O_z(\alpha,\beta) \overset{t'}{\rightsquigarrow} (x,x)] \cdot c^{l(t')+1} \\
&= c \sum_{z=1}^{|O((\alpha,\beta))|} \sum_{t':O_z((\alpha,\beta))\rightsquigarrow(x,x)} P((\alpha,\beta) \to O_z(\alpha,\beta)) \cdot P[t'] \cdot c^{l(t')} \\
&= c \sum_{z=1}^{|O((\alpha,\beta))|} P((\alpha,\beta) \to O_z(\alpha,\beta)) \sum_{t':O_z((\alpha,\beta))\rightsquigarrow(x,x)} P[t'] \cdot c^{l(t')} \\
&= c \sum_{z=1}^{|O((\alpha,\beta))|} P((\alpha,\beta) \to O_z(\alpha,\beta)) \cdot s'(O_z(\alpha,\beta)) \\
&= c \sum_{i=1}^{|O(\alpha)|} \sum_{j=1}^{|O(\beta)|} P((\alpha,\beta) \to (O_i(\alpha), O_j(\beta))) \cdot s'(O_i(\alpha), O_j(\beta)) \\
&= c \sum_{i=1}^{|O(\alpha)|} \sum_{j=1}^{|O(\beta)|} \frac{w(\alpha, O_i(\alpha)) \cdot w(\beta, O_j(\beta))}{W_O(\alpha) \cdot W_O(\beta)} \cdot s'(O_i(\alpha), O_j(\beta)) \\
&= \frac{c}{W_O(\alpha) \cdot W_O(\beta)} \sum_{i=1}^{|O(\alpha)|} \sum_{j=1}^{|O(\beta)|} w(\alpha, O_i(\alpha)) \cdot w(\beta, O_j(\beta)) \cdot s'(O_i(\alpha), O_j(\beta))
\end{aligned}
\tag{C.2}
$$

Equation C.2 is identical to the SimRank equation 4.11 with $c = C$ and in-edges swapped for out-edges. Since the solution to 4.11 is unique[1], $s'(\alpha, \beta) = s(\alpha, \beta)$ for all $\alpha, \beta \in V$.

[1]Straight forward proof on lines of SimRank. Refer [**?**]

Appendix D

Personalized SimRank Properties

In this appendix, we prove some of the important properties of the Personalized SimRank System introduced in Section 4.3.2.

PROPOSITION 0 : The sequence $S_k(\alpha, \beta)$ converges for every node pair $\alpha, \beta \in V$

PROOF : It follows from the monotonocity of the sequence $S_k(\alpha, \beta)$:

$$0 \leq S_k(\alpha, \beta) \leq S_{k+1}(\alpha, \beta) \leq 1 \text{ for all } \alpha, \beta \in V, k \geq 0 \tag{D.1}$$

This says for every α, β, the sequence $\{S_k(\alpha, \beta)\}$ is bounded and non-decreasing as k increases and hence each sequence $\{S_k(\alpha, \beta)\}$ converges to a limit $S(\alpha, \beta) \in [0, 1]$.

PROPOSITION 1 : The system of equations of Personalized SimRank has a solution, which is constructed by the iterative fixed point solution defined in Section 4.3.2.

PROOF : The unique solution is actually constructed in Section 4.3.2, and the correctness of the iterative algorithm follows.

Using PROPOSITION 0 and the fact that limit of a sum is the sum of the limits,

we have :

$$S(\alpha,\beta) = \begin{cases} 1 & \text{if } \alpha = \beta \\ InitStore[(\alpha,\beta)] & \text{if } (\alpha,\beta) \in InitStore \\ \frac{C}{W_I(\alpha)W_I(\beta)} \sum_{i=1}^{|I(\alpha)|} \sum_{j=1}^{|I(\beta)|} w(I_i(\alpha),\alpha) \cdot w(I_j(\beta),\beta) \cdot S(I_i(\alpha),I_j(\beta)) & \text{otherwise} \end{cases} \tag{D.2}$$

which shows that the limits $S(*,*)$ satisfy the simrank equations.

PROPOSITION 2 : The system of equations of Personalized SimRank has a unique solution.

PROOF : Suppose $s_1(*,*)$ and $s_2(*,*)$ are two solutions to the $|V|^2$ Personalized SimRank equations. For all $\alpha, \beta \in V$, let $\delta(\alpha,\beta) = s_1(\alpha,\beta) - s_2(\alpha,\beta)$ be their difference. Let $M = max_{(\alpha,\beta)}|\delta(\alpha,\beta)|$ be the maximum absolute value of any difference. We need to show that $M = 0$. Let $|\delta(\alpha,\beta)| = M$ for some $\alpha, \beta \in V$. Certainly $M = 0$ if α, β or if $(\alpha,\beta) \in InitStore$ or if $I(\alpha) = \emptyset$ or $I(\beta) = \emptyset$. Otherwise, $s_1(\alpha,\beta)$ and $s_2(\alpha,\beta)$ are the weighted average scores of their in-neighbours. That is, from equation 4.12,

$$\begin{aligned} s_1(\alpha,\beta) =& \frac{C}{W_I(\alpha)W_I(\beta)} \sum_{i=1}^{|I(\alpha)|} \sum_{j=1}^{|I(\beta)|} w(I_i(\alpha),\alpha) \cdot w(I_j(\beta),\beta) \cdot s_1(I_i(\alpha),I_j(\beta)) \\ s_2(\alpha,\beta) =& \frac{C}{W_I(\alpha)W_I(\beta)} \sum_{i=1}^{|I(\alpha)|} \sum_{j=1}^{|I(\beta)|} w(I_i(\alpha),\alpha) \cdot w(I_j(\beta),\beta) \cdot s_2(I_i(\alpha),I_j(\beta)) \end{aligned}$$

In terms of $\delta(\alpha,\beta)$,

$$\begin{aligned} \delta(\alpha,\beta) =& s_1(\alpha,\beta) - s_2(\alpha,\beta) \\ =& \frac{C}{W_I(\alpha)W_I(\beta)} \sum_{i=1}^{|I(\alpha)|} \sum_{j=1}^{|I(\beta)|} w(I_i(\alpha),\alpha) \cdot w(I_j(\beta),\beta) \cdot [s_1(I_i(\alpha),I_j(\beta)) - s_2(I_i(\alpha),I_j(\beta))] \\ =& \frac{C}{W_I(\alpha)W_I(\beta)} \sum_{i=1}^{|I(\alpha)|} \sum_{j=1}^{|I(\beta)|} w(I_i(\alpha),\alpha) \cdot w(I_j(\beta),\beta) \cdot \delta(I_i(\alpha),I_j(\beta)) \end{aligned}$$

Thus,

$$\begin{aligned}
M =& |\delta(\alpha,\beta)| \\
=& \left| \frac{C}{W_I(\alpha)W_I(\beta)} \sum_{i=1}^{|I(\alpha)|} \sum_{j=1}^{|I(\beta)|} w(I_i(\alpha),\alpha) \cdot w(I_j(\beta),\beta) \cdot \delta(I_i(\alpha), I_j(\beta)) \right| \\
\leq& \frac{C}{W_I(\alpha)W_I(\beta)} \sum_{i=1}^{|I(\alpha)|} \sum_{j=1}^{|I(\beta)|} w(I_i(\alpha),\alpha) \cdot w(I_j(\beta),\beta) \cdot |\delta(I_i(\alpha), I_j(\beta))| \\
\leq& \frac{C}{W_I(\alpha)W_I(\beta)} \sum_{i=1}^{|I(\alpha)|} \sum_{j=1}^{|I(\beta)|} w(I_i(\alpha),\alpha) \cdot w(I_j(\beta),\beta) \cdot M \\
=& CM
\end{aligned}$$

Since $0 < C < 1$, it follows that $M = 0$

PROPOSITION 3 : *The difference between SimRank theoretical and iterative similarity scores decreases exponentially in the number of iterations for every pair of nodes. Precisely, for every iteration number $t = 0, 1, 2, \ldots$ and for every two nodes α, β, the following estimate holds:*

$$s(\alpha,\beta) - S_t(\alpha,\beta) \leq C^{t+1} \tag{D.3}$$

PROOF : If $\alpha = \beta$ then $s(\alpha,\alpha) = S_k(\alpha,\alpha) = 1$ by definition for every $k = 0, 1, 2, \ldots$, the left side of D.3 is 0 and thus D.3 obviously holds. Similarly, if $I(\alpha) = \emptyset$ or $I(\beta) = \emptyset$ then by definition $s(\alpha,\beta) = S_k(\alpha,\beta) = 0$. Also, lets consider the case when $(\alpha,\beta) \in InitStore$. In this case, $s(\alpha,\beta) = S_k(\alpha,\beta) = InitStore[(\alpha,\beta)]$ and hence D.3 always holds.

For the general case of $\alpha \neq \beta$, $I(\alpha) \neq \emptyset$, $I(\beta) \neq \emptyset$ and $(\alpha,\beta) \in InitStore$, the proof is organized by mathematical induction.

Induction Basis : Let us prove that D.3 holds for $k = 0$ i.e. for every two

nodes α, β :

$$s(\alpha, \beta) - S_0(\alpha, \beta) \leq C \tag{D.4}$$

Since $\alpha \neq \beta$, $I(\alpha) \neq \emptyset$, $I(\beta) \neq \emptyset$ and $(\alpha, \beta) \in InitStore$, then $S_0(\alpha, \beta) = 0$.

$$\begin{aligned}
s(\alpha, \beta) - S_0(\alpha, \beta) &= s(\alpha, \beta) \\
&= \frac{C}{W_I(\alpha) W_I(\beta)} \sum_{i=1}^{|I(\alpha|} \sum_{j=1}^{|I(\beta)|} w(I_i(\alpha), \alpha) \cdot w(I_j(\beta), \beta) \cdot \underbrace{s(I_i(\alpha), I_j(\beta))}_{\leq 1} \\
&= \frac{C}{W_I(\alpha) W_I(\beta)} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(\beta)|} w(I_i(\alpha), \alpha) \cdot w(I_j(\beta), \beta) \\
&\leq C
\end{aligned}$$

which proves D.4

Inductive Step : Provided that D.3 holds for a given k for all node pairs, let us prove that D.3 holds for $(k+1)$ as well:

$$\begin{aligned}
s(\alpha, \beta) - S_{k+1}(\alpha, \beta) &= \frac{C}{W_I(\alpha) W_I(\beta)} \sum_{i=1}^{|I(\alpha)|} \sum_{j=1}^{|I(\beta)|} w(I_i(\alpha), \alpha) \cdot w(I_j(\beta), \beta) \cdot s(I_i(\alpha), I_j(\beta)) \\
&\quad - \frac{C}{W_I(\alpha) W_I(\beta)} \sum_{i=1}^{|I(\alpha)|} \sum_{j=1}^{|I(\beta)|} w(I_i(\alpha), \alpha) \cdot w(I_j(\beta), \beta) \cdot S_k(I_i(\alpha), I_j(\beta)) \\
&= \frac{C}{W_I(\alpha) W_I(\beta)} \\
&\quad \sum_{i=1}^{|I(\alpha)|} \sum_{j=1}^{|I(\beta)|} w(I_i(\alpha), \alpha) \cdot w(I_j(\beta), \beta) \cdot \underbrace{[s(I_i(\alpha), I_j(\beta)) - S_k(I_i(\alpha), I_j(\beta))]}_{\leq C^{k+1} \text{ by inductive hypothesis}} \\
&\leq C^{(k+1)+1}
\end{aligned}$$

The latter finally proves D.3.

Appendix E

Probabilistic Outputs for SVM

The following is the improved code by [**?**] for the algorithm proposed originally by [**?**], that theoretically converges and avoids numerical difficulties.

```
Input parameters:
  deci = array of SVM decision values
  label = array of booleans: is the example labeled +1?
  prior1 = number of positive examples
  prior0 = number of negative examples
Outputs:
  A, B = parameters of sigmoid

//Parameter setting
maxiter=100
//Maximum number of iterations
minstep=1e-10
//Minimum step taken in line search
sigma=1e-12
//Set to any value > 0
//Construct initial values: target support in array t,
//initial function value in fval
```

```
hiTarget=(prior1+1.0)/(prior1+2.0), loTarget=1/(prior0+2.0)
len=prior1+prior0 // Total number of data
for i = 1 to len {
    if (label[i] > 0)
        t[i]=hiTarget
    else
        t[i]=loTarget
}
A=0.0, B=log((prior0+1.0)/(prior1+1.0)), fval=0.0
for i = 1 to len {
    fApB=deci[i]*A+B
    if (fApB >= 0)
        fval += t[i]*fApB+log(1+exp(-fApB))
    else
        fval += (t[i]-1)*fApB+log(1+exp(fApB))
}
for it = 1 to maxiter {
    //Update Gradient and Hessian (use H = H + sigma I)
    h11=h22=sigma, h21=g1=g2=0.0
    for i = 1 to len {
        fApB=deci[i]*A+B
        if (fApB >= 0)
            p=exp(-fApB)/(1.0+exp(-fApB)), q=1.0/(1.0+exp(-fApB))
        else
            p=1.0/(1.0+exp(fApB)), q=exp(fApB)/(1.0+exp(fApB))
        d2=p*q
        h11 += deci[i]*deci[i]*d2, h22 += d2, h21 += deci[i]*d2
        d1=t[i]-p
        g1 += deci[i]*d1, g2 += d1
```

```
    }
    if (abs(g1)<1e-5 && abs(g2)<1e-5) //Stopping criteria
        break
    //Compute modified Newton directions
    det=h11*h22-h21*h21
    dA=-(h22*g1-h21*g2)/det, dB=-(-h21*g1+h11*g2)/det
    gd=g1*dA+g2*dB
    stepsize=1
    while (stepsize >= minstep){ //Line search
        newA=A+stepsize*dA, newB=B+stepsize*dB, newf=0.0
        for i = 1 to len {
            fApB=deci[i]*newA+newB
            if (fApB >= 0)
                newf += t[i]*fApB+log(1+exp(-fApB))
            else
                newf += (t[i]-1)*fApB+log(1+exp(fApB))
        }
        if (newf<fval+0.0001*stepsize*gd){
            A=newA, B=newB, fval=newf
            break //Sufficient decrease satisfied
        }
        else
            stepsize /= 2.0
    }
    if (stepsize < minstep){
        print Line search fails
        break
    }
}
```

```
if (it >= maxiter)
    print Reaching maximum iterations
return [A,B]
```

Bibliography

David D Lewis. Text representation for intelligent text retrieval: a classification-oriented view. *Text-based intelligent systems: current research and practice in information extraction and retrieval*, pages 179–197, 1992.

David D Lewis and Marc Ringuette. A comparison of two learning algorithms for text categorization. In *Third annual symposium on document analysis and information retrieval*, volume 33, pages 81–93, 1994.

Isabelle Moulinier, Gailius Raskinis, and J Ganascia. Text categorization: a symbolic approach. In *proceedings of the fifth annual symposium on document analysis and information retrieval*, pages 87–99, 1996.

J Ross Quinlan. Learning efficient classification procedures and their application to chess end games. In *Machine learning*, pages 463–482. Springer, 1983.

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

Gerard Salton and Chung-Shu Yang. On the specification of term values in automatic indexing. *Journal of documentation*, 29(4):351–372, 1973.