Word Similarity Modelling with Slot Phrases

Muralikrishna.S

A DISSERTATION
in
Department of Computer Science,
Royal Holloway, University of London,

for

Masters by Research in Machine Learning

22nd October 2004

Abstract

Statistical language learning and language processing generally needs some representation of the semantic and syntactic characteristics of words. This thesis looks at describing a word by its statistical pattern of use in real language.

Starting with little linguistic prior knowledge, we use linear algebra to represent words as points in a low dimensional Euclidean space. We then conduct exploratory analysis of this “word space” representation and show that it does indeed capture significant semantic and syntactic properties of words.

We conclude that this representation may be a useful starting point for automatic induction of word meaning and higher language modeling.

Contents

Acknowledgements 4

1 Word Similarity Models 5

1.1 Introduction 5

1.1.1 Similarity Models and their applications 5

1.1.2 Strategies for Construction of Similarity models – Knowledge Rich and Knowledge Poor Techniques 6

1.1.3 A Philosophical Basis for Knowledge Poor Techniques 6

1.2 The Objective 7

1.3 Related Work on Word Similarity Models 8

1.3.1 Windows Based Approach 9

1.3.2 Syntax Based Approach 14

1.3.3 Summary 17

2 Slot Phrases 19

2.1 Introduction 19

2.1.1 Slot Phrases 20

2.1.2 Some distinguishing features of slot phrases 21

2.2 Practical considerations in using slot phrases to define contexts 22

2.2.1 Criteria for Identification of Phrases 22

2.2.2 Criteria for the selection of words to be replaced by slots 23

2.3 Practical steps towards constructing contexts using slot phrases 24

2.4 Low Dimensional Representation using SVD 28

3 Introduction to machine learning methods used in this work 29

3.1 Introduction 29

3.2 Singular Value Decomposition 29

3.2.1 Introduction to Singular Value Decomposition 29

3.2.2 Mathematical Proof of Singular Value Decomposition 30

3.2.3 Synthetic Experiments to Illustrate the working of SVD . . . 34
3.3 Clustering . . . 37
3.3.1 k-means Clustering . . . 38
3.4 The Support vector machine . . . 38
3.4.1 Introduction . . . 38
3.4.2 The classification problem and primal formulation of the SVM . . . 40
3.4.3 Same optimal values for the primal and the dual form 41
3.4.4 The SVM dual form . . . 43
3.4.5 Incorporating Kernels in the Dual Form . . . 45
3.5 Canonical Correlation Analysis . . . 46
3.6 Mathematical Derivation of KCCA . . . 46

4 Experiments on similarity modelling with slot phrases 51
4.1 Introduction . . . 51
4.2 From Slot-Word matrix to Word Space using SVD . . . 52
4.2.1 Introduction . . . 52
4.2.2 Practical steps to obtain the word space using SVD . 53
4.3 Finding Semantic and Syntactic Information in the Lower Dimensional Word Space: A summary . . . 55
4.3.1 Introduction . . . 55
4.4 Exploratory analysis of the word space . . . 56
4.4.1 Nearest Neighbors for Selected Words in the general slots word space . . . 56
4.4.2 Clustering in the Word Space . . . 62
4.4.3 Using SVM to find similar words . . . 66
4.4.4 Experiments with specific slot phrases . . . 68
4.5 Comparison with Schütze's word space . . . 74
4.5.1 Comparison using nearest neighbors . . . 75
4.5.2 Comparison using KCCA . . . 76

5 Discussion and future work 80

List of Tables

3.1 The Original Data Matrix A. Note: (a) C.U=Commented upon (b)The morphological variations such as inferred (of infer) are considered the same as their root words in forming the slot-word matrix. (c) SP-refers to slot phrases. 36
3.2 The reconstructed Matrix-$A_2 = U_2 S_2 V_2'$ of A Using 2 eigen values and vectors 36

4.1 Number of similar words to 5 words for each value of k 55
4.2 The nearest neighbors for 5 random words and their syntactic forms. 59
4.3 Nearest neighbors for very common words 61
4.4 Nearest neighbors for typical categories 61
4.5 Some typical semantic clusters 64
4.6 Some typical syntactic clusters 65
4.7 Classification results of the SVM 67
4.8 The nearest neighbors resulting from general(G) and specific(S) slot phrases for 10 high frequency words. 72
4.9 10 Nearest neighbours for 5 rare words. The `font` is used to mark semantically *very* close words. 75
4.10 The nearest neighbors resulting from using Specific Slot phrases(labelled as 'specific SP') and using Schütze's technique(labelled as 'Schütze' for 10 high frequency words. 77
4.11 The first row lists the first 10 canonical correlations obtained from KCCA of Schütze's word space(Sch) and the word space obtained using general slot phrases(GSP). The second row lists the correlations for Schütze's space(Sch) and the word space obtained using specific slot phrases(SSP). 78

Acknowledgements

First and foremost I would like to thank my supervisor Chris Watkins. Chris deserves a great deal of thanks for his constant encouragement and support. His ideas have been fundamental to this work and his suggestions have been instrumental in my experiments. I have enjoyed interacting with him and this interaction has greatly benefited me.

The Department of Computer Science at Royal Holloway has been a very friendly Research environment and I am thankful to its members. I would like to particularly thank Gill Adourian and Janet Hales both of whom have been extremely friendly and helpful to me. I would like to thank Z Luo for his interest and encouragement.

Finally, I would like to thank my Mother, Father and Grandmother, who have supported me in every way to pursue this course. My fiance Bhavani deserves special thanks, as she has inspired me to work hard and do well. My biggest thanks goes to my Grandfather, who as a well known scientist has always inspired me towards pursuing research.

Chapter 1

Word Similarity Models

1.1 Introduction

1.1.1 Similarity Models and their applications

Over recent years, many models of word similarity have been developed for several applications. These models have been used for applications such as language modelling, automatic thesaurus construction and automatic grammar induction.

One such application is speech recognition where similarity models are used for language modelling. In speech recognition, hidden markov models combine linguistic evidence and speech evidence to predict a word from the previous words that have been spoken. The HMM uses cooccurrence probabilities from a training set to construct a language Markov model from which it draws evidence. For instance, it would be needed to predict whether *two* or *too* follows *I was* in the sentence "I was (*two/too* happy)". However coocurrence probabilities for many word pairs(or triplets) may not be available even in a huge training set. This problem, common to many applications, is known as the *sparse data problem.*

Similarity Models are able to resolve the sparse data problem by finding the most similar seen words to the unseen words. The coocurrence probabilities of the encountered words are then used as an estimate to predict the probabilities of an unseen coocurrence. Other uses of word similarity models in language modelling are machine translation and word sense disambiguation

Another application of word similarity models is in automatic thesaurus construction. Automatically constructed thesauruses have distinct advantages over manual ones. They are adaptable to change in language over time, as language being productive constantly undergoes change. Moreover they are not prone to the bias that may arise in human-constructed thesauri. These automatic thesauri have many applications such as semantic parsing and query reformulation for search engines.

Finally, word similarity models are used in research on automatic grammar induction from text. Word similarity models which automatically capture syntactic and semantic similarities can be used to induce grammar automatically from a text corpus. Automatic grammar induction is more adaptable to changes in language style (over time) than manually constructed grammars. Moreover, while part of speech labels are available for English, such information is not available for less common languages. [16]

1.1.2 Strategies for Construction of Similarity models – Knowledge Rich and Knowledge Poor Techniques

The strategies for building such word-similarity models can be roughly divided into two categories: knowledge-rich and knowledge-poor methods, according to the amount of knowledge they presuppose (Grefenstette 1995) [4]

Knowledge-rich methods require some sort of previously encoded information such as domain-knowledge-structures, or resources such as hand crafted thesaurus like Word Net and Roget's thesaurus. However such methods suffer from many limitations such as limited vocabulary size, unclear classification criteria and considerable time and effort required in building a thesaurus by hand.

On the other hand, knowledge poor methods use no presupposed semantic knowledge for automatically extracting syntactic/semantic information. They use the frequency of coocurrence of words within various contexts to compute semantic similarity among words. These models are based on a natural assumption that two *words can be considered similar if they tend to occur in similar contexts.* Using this hypothesis, several researchers have developed different contexts for building similarity models.

1.1.3 A Philosophical Basis for Knowledge Poor Techniques

A philosophical justification for using knowledge poor methods to derive meaning(semantic relations) of words can be found in the Witgensteinian view of language. According to this view, the meaning of a word is *completely manifested and founded* in linguistic praxis or *language use.* Put in another way, the meaning of "meaning" must be answered "from within" a theory of language, since words do not (and in a stronger sense *cannot*) have meaning outside language. This is expressed in the famous dictum "meaning is use"(Magnus Sahlgren 2002). This theory then backs knowledge-poor methods, according to which it is possible to derive to the meaning of a word from its *use* in the language.

The use of contexts by knowledge poor methods to model meaning, derives its basis from the distributional hypothesis by Zellig Harris who in this book "Mathematical Structures of Language"(1968) states that " ..the meaning of entities , and the meaning of grammatical relations among them, is related to the restriction on combination of these entities relative to other entities." These combinational restrictions can be viewed as the constraints that the *contexts* of a word imposes on a word.

According to this view, the *use* of a word is manifested in the totality of constraints with all other words, in other words, by its *distribution* in various contexts. Since the *meaning* of a word is determined by its *use* in language, it can be said that the *meaning* of word can be derived from the *contexts* in which it is distributed. In summary, it can be said that two words are related in meaning if they happen to be distributed similarly or if they share the same contexts.

It is important to note that the "context" which is the totality of all the relationships which the word participates in, may be hard to model. Moreover, "language use" often does not provide the whole context from which the meaning of a word may be inferred, in the sense that "Everything is not said".

Despite these limitations, modelling word similarity using contexts has been able to produce impressive results for various applications such as information retrieval, automatic thesaurus construction and language modelling.

1.2 The Objective

The rest of the thesis is motivated by the above idea of using *contexts* to construct word-similarity models. The goal of this project is to introduce **Slot Phrases** as an alternative context for word similarity models.

In the next section, I have reviewed the previous research in word-similarity modelling. Chapter 2 introduces and develops the idea of slot phrases for modelling word similarity. Chapter 3 describes four machine learning methods that I have used in this work. They are Singular value decomposition(SVD), Support vector machines(SVM), K-means clustering and kernel canonical correlation analysis(KCCA).

Chapter 4 describes the construction of the low dimensional vectorial representation of words and several exploratory investigations to analyse this representation. This chapter also describes an experiment that evaluates the representation in the context of previous work in this area of research.

Chapter 5 summarizes the work described in this thesis with several suggestions on how the present work can be extended in the future.

1.3 Related Work on Word Similarity Models

Three main research issues can be identified from a study of previous research in constructing word-similarity models:

- First is the *context*. Broadly speaking it is possible to distinguish two kinds of contexts which have been used by the previous researchers: *Window Based Contexts* and *Syntax Based Contexts*. Window based techniques consider a window of an arbitrary number of words around the word as a context for that word. On the other hand, syntax based approaches consider syntactic relationships as possible contexts.

- The other area of research is whether the model should be a *class based* model or a *similarity based* model. This decision becomes important for applications where the sparse data problem arises[1].

  In the class based approach the probablity of an unseen word coocurrence such as $word_1 - word_2$ is calculated using probablities of observed coocurences such as $word - word_2$, where, $word$ belongs to the same class as $word_1$.

  On the other hand, similarity based models calculate the probablity of an unseen coocurrence $word_1 - word_2$, using probablities of observed coocurrences such as $word - word_2$, where $word$ is among the words most similar to $word1$. Similarity based techniques use search techniques to find the most similar words to a given word [8].

- The third area of research is the *method* for computing similarity.

I have reviewed the previous research keeping in mind these three issues. However, the work of previous researchers have been discussed within the broad division of *windows based* and *syntax based* approaches.

[1]Sparse data problem was introduced in section 1.1.1 needs to be solved

1.3.1 Windows Based Approach

Introduction

The windows based approach primarily derived inspiration from the Information Retrieval community where, the aim is to retrieve relevant documents given some key-words which are characteristic of those documents. In the information retrieval task, the documents are first represented as a bag of words (some pre-processing may be performed to select important words). Then a document-by-word matrix is formed, where each cell(i, j) contains the frequency of a $word_j$ in $document_i$. Thus each document is represented in a vector space in terms of the words they contain.

In order to find the most similar documents to a given document or a given query, these (documents or queries) are first represented in terms of the words in the *document-by-word matrix.* Singular value decomposition (SVD) is often used to obtain a lower dimensional representation of the document-by-word matrix. Documents are represented as low dimensional vectors according to this representation.

The low dimensional vectorial representation enables easier computation of similarity between documents or between documents and word vectors. Moreover this representation is expected to capture better relationships between the documents than the corresponding high-dimensional representation. Therefore, this representation has been used by the information retrieval community for retrieving documents most similar to a given document or a word vector.

Windows-based methods follow a very similar technique for constructing word similarity models. Here, words are represented in a word-by-context matrix, where the contexts are windows around the words. Cell(i, j) contains the frequency of a $word_i$ in $window_j$. These windows can be as large as document or just the neighboring word. As two similar words are expected to share similar *contexts vectors*, similarity can computed in this space using various similarity metrics available.

This approach of using windows as a context can be justified using the following two intuitively appealing assumptions (Derrick Higgins 2002) [5]:

- **Proximity Assumption**: Similar words tend to occur near each other. For example in the following sentence: the market sold, *apple, banana* and *oranges* at a discounted rate. Here the words *apples, bananas* and *oranges* occur in proximity.

- **Topicality assumption**: Similar words tend to have the same neighboring content words[2] This is called the *topicality assumption* as similar words tend to appear in the same *topic contexts* even though they may not occur near each other.

Experiments using the windows based approach can be divided into two subcategories: Experiments with *larger windows* and experiments with *smaller windows.* Such a distinction is necessary as they capture different kinds of information. This is discussed further after a review of experiments on larger and smaller windows.

Experiments with Larger Windows

- In 1992, Schütze [15] used a window size of 1000 characters to capture a neighborhood for a word. Using a context-word matrix similar to the word-document matrix, he performed SVD to obtain a lower dimensional representation of these words. Various similarity measures such as cosine and Euclidean distance were used to obtain words similar to a given word in this space. However it was observed that this method does not succeed in producing sharp semantic classes needed for applications such as lexical acquisition.

- In 1998, T.K.Landauer et al. [22] represented the text data as a words-by-documents coocurrence matrix (60000 $\times$ 30000) in which each cell indicates the frequency of a given word in a text sample of 150 words. They use SVD to obtain a more compact representation of a word in terms of 300 dimensions. The resulting similarity model was tested using a synonym test called TOEFL ("Test of English as a foreign Language). For each word four alternatives are given, and the contestant is asked to find the one that's the most synonymous. Choosing at random would yield 25 % correct. However, the model performed correctly in 64 % of the cases. The results using the original 30,000 dimensional vectors (before SVD) were not nearly as good : only 36 % correct. It was therefore concluded that SVD discovers latent relationships similar to the way human beings do.

- Kanerva et al(2000) [9] point out that SVD is computationally heavy for a huge matrix and is impractical for a collection of a million documents. They use a method called random labelling to obtain sig-

[2]Content words are usually referred to words other than the closed class of words. Closed class of words are words which have grammatical functionality rather than much semantic content. Examples are prepositions, conjunctions etc.

nificantly smaller matrices (60000×1800 dimensions) rather than the 60000×30000 dimensionsional matrix used in SVD (from documents of 150 words each). Instead of collecting the word coocurrence statistics in a 60000×30000 matrix, they construct a lower dimensional matrix(60000×1800) G as follows:

1. Set $G = 0$.
2. For each document d, make an N-dimensional (e.g. 1800 Dimensional) ternary Index Vector i_d by placing a small number k (e.g.10) of -1s and +1s at random among the 4,000; the rest will be 0s:
   For example, $i_d = 0000000(+1)000...000(+1)000000(-1)000...000$, that is, 4000 "trits"(ternary digits) and 3980 of them being 0's.
3. Scan through document each document. Every time the word w appears in document d, add the index vector i_d to row w of matrix G.

They mathematically justify the above procedure proving that the matrix G approximately conveys the same amount of information as the document-by-word matrix.

The results reported on the TOEFL test for a large window of 150 words was around 48-51% and as high as 62-70% on a smaller window.

Experiments with Smaller Windows

- Lund and Burgess(1996) [11] represented the text data as a *words-by-words* coocurence matrix, instead of the words-by-windows matrix. Each $cell_{i,j}$ in this matrix indicated the co-ocurance frequency of $word_i$ and $word_j$, within a sliding 8 word window. According to their definition of coocurence, words within this window are recorded as coocuring with a strength inversely proportional to the number of words separating then within that window.

  By this representation, each word is represented both by row(w_i, w_j) and column (w_j, w_i) vectors. Therefore, the $n \times n$ matrix and its transpose may be concatenated to produce a $n \times 2n$ matrix in which each word is now represented by a $2n$ sized vector. They obtain a lower dimensional representation of the word vectors by taking only a projection of the data on the top principal components of the data matrix.

The results show that the nearest neighbors of words are closely related in meaning. Moreover it was observed that the performance of the system diminishes when the window size is made greater than 8 words.

- Schütze(1997) [17] takes a similar approach by representing the text data as a words-by-words coocurrence matrix with coocurrence frequencies within a window of 40 words. He uses SVD to obtain a compact representation in terms of 20 dimensions. In this paper it is argued that local coocurrence statistics are more informative than document based coocurence statistics.(Magnus Sahlgren 2002 [14]).

- Schütze(1993), in another experiment [16] uses a 4 word neighborhood in a unique way to model *syntactic word similarity.* Four nearest positions (-2,-1,+1 and +2) to a word on either side of a word are used to form four word-by-word coocurrence matrices. These matrices contain coocurrences frequencies in these four positions with respect to a word. He concatenates these four matrices. By adopting such a technique he achieves syntactic clustering.

  He argues that the immediate left position (-1) is a good indicator of an adjective or noun, (-2) for a noun. For example, in many simple sentences, verbs generally occur 1 word after the noun (eg. "the dog barked") etc. However Schütze does away with the closed words and only the content words are taken into consideration in his experiments.

- Magnus Sahlgren(2002) [14] uses random labelling of words, similar to Kanerva's [9] method, in very narrow context windows (2,4,6 and 8 words) to calculate the semantic *context vectors* for each word type in the text data. His results (65.6%) on the TOEFL test indicate that random indexing produces results that is equivalent and some times exceeds in performance compared to LSA(64.5%), especially when using narrow contexts. The results show a further increase when grammatical information is used.

- Ido Dagan, Marcus and Markovitch [8] (1993 and 1995) constructed a similarity based model using a 6 word window for a context. They questioned the efficacy of class based models built by other researchers before them, and constructed a similarity based model.

  In contrast to other windows based approaches which use the words-by-windows matrix or the words-by-words matrix, their approach used pointwise mutual information in computing similarity between words.

In their similarity based model, the mutual information between two words such as (chapter,describes) of which one word (eg. chapter) does not occur in the corpus, is computed as the average of the mutual information of the most similar pairs ((book,describes),(introduction,describes)..etc) that are present in the corpus.

A similarity network is built by calculating pairwise similarities between words. Similarity is calculated in two steps. First, *point wise mutual information*, which measures the strength of contextual association, is calculated for pairs of words (w, w_1), in the same context, as follows:

$$I(w_1, w) = log\frac{P(w_1, w)}{P(w_1)P(w)} = log\frac{\frac{f(w_1,w)}{N}}{\frac{f(w_1)}{N}\frac{f(w)}{N}}$$

Here $f(w_1, w)$ is the frequency of joint occurrence of the words w_1 and w and $f(w_1)$ and $f(w)$ are the frequencies of their occurrences in the text. N is the total number of contexts. Due to the unreliability of measuring negative mutual information, any negative value is assigned the value 0. $I(w_1, w)$ is also set to 0, if $f(w_1, w) = 0$.

Using this the similarity of two words w_1, w_2 is calculated as follows.

$$sim(w_1, w_2) = \frac{\sum_w (min(I(w, w_1), I(w, w_2)) + min(I(w_1, w), I(w_2, w)))}{\sum_w (max(I(w, w_1), I(w, w_2)) + max(I(w_1, w), I(w_2, w)))}$$

However, their model does not provide probability estimates and disambiguating decisions (in applications) were based on comparing scores for different alternatives. Therefore it could not be used as a part of a larger probabilistic model needed for applications like speech recognition. In 1997 Dagan, Lee and Pereira [7] extend the above work in producing explicit probability estimates.

Larger vs Smaller Windows: a comparison

It has been argued that smaller windows yield more information about the meanings of the words than larger windows. Magnus Sahlgren (2002) [14] argues that “a more sizable context might give better clues to what a particular word is *about* than to what it *means*. The idea is that two words that *are about* similar things will occur in similar *regions* (eg. documents), while two words that have similar meanings will occur in similar context *neighbors* (i.e. words)”. This implies that larger contexts would be more suited to tasks such as information retrieval where topical information is more useful, whereas smaller contexts would find greater applicability in lexical and language modelling tasks.

1.3.2 Syntax Based Approach

A number of researchers have used syntactic relations for context. These have been inspired by the computational linguistics community who were studying problems such as modelling word coocurrence (for speech recognition), sense disambiguation and syntactic parsing of sentences. The appearance of reliable parsers (and taggers) made it possible to automatically discover words in syntactic relationships (Subj-Obj, Noun-Adj etc.) from a corpus. Thus, word co-ocurance statistics within syntactic relationships could be automatically collected from a text corpus. Word similarity models could then be constructed using these syntactic relationships as possible contexts.This is reffered to as the *syntax based approach.*

The common thread in most syntax based approaches is to first compute a information theoretic measure-ITM[3](w, c) of words w with their contexts c and then calculate the similarity between words w_1 and w_2 using ITM(w_1, c) and ITM(w_2, c).

The syntax based approach is based on what has been called the "parallelism assumption" by Derick Higgins 2002 [5] according to which similar words tend to occur in similar grammatical relationship with other words. Several researchers have based their models on this assumption. The following paragraphs describe some of the works of previous researchers who have used the syntax based approach.

- An early attempt to automatically classify words into semantic classes using noun-verb relationship was carried out by Grishman et al. in 1986.

- In 1990 Hindle [6] developed a similarity model using a method to group nouns according to the *verb contexts* they share. His work is based on the idea that similar nouns would share similar verb contexts. He formalizes this idea in three steps. First he relates verbs and nouns by computing pointwise mutual information between pairs of verbs and nouns.

  Next, he takes pointwise similarity between two nouns(n_1, n_2) with respect to a verb(v_1) as a minimum of $I(n_1, v_1)$ and $I(n_2, v_2)$. These computations are done separately for the nouns appearing as subject

[3]ITM stands for information theoretic measure

and object respectively. Finally, he finds the overall similarity between two nouns by summing up the pointwise similarity across all the verbs for both the subject and object case.

$$sim(n_1, n_2) = \sum_v min(I(v, n_1), I(v, n_2)) + min(I(n_1, v), I(n_2, v))$$

He evaluates the results by examining the nearest neighbors to sample sets of words. According to his evaluation for many nouns, semantically similar nouns are found as nearest neighbors, although for some nouns such semantically similar neighbors are not found.

- Pereira [2] et.al. extended Hindle's approach of using subj-verb-obj as a context to model similarity between nouns. However unlike Hindle they constructed a *soft-class based model* by clustering nouns into classes. They construct these classes by an iterative algorithm (like the EM algorithm) to find the cluster centroids. This is carried out in two steps. First they estimate the cluster centroids given fixed membership (of nouns to these clusters) probabilities. Second, the entropy between the data and the model is minimized. This way, through an iterative process, the centers of the clusters are found. In this experiment, they build hierarchical clusters and it was concluded that in many cases the clusters are intuitively informative.

- Lin(1998) [10] used a broad-coverage parser to extract various types of relationships between the words in a sentence. For example, the triplets extracted from "I have a brown dog" are :
(have subj I)(I subj-of have)(dog obj-of have)(dog adj-mod brown)(brown adj-mod-of dog)(dog det a)(a det-of dog)
These triplets are generalized using the frame: (w, r, w'), where r stands for the grammatical relationship between w and w'. According to the notation he uses, $\|w, r, w'\|$ represents the frequency of the tupule (w, r, w') . To represent other patterns he uses the wildcard (*). For instance, $\|w, r, *\|$ represents the total co-ocurances of w and r in the parsed corpus. Using this information, he derives that the mutual information of two words w and w' with respect to a relation r is:
$$I(w, r, w') = \log\left(\frac{\|w, r, w'\|\|*, r, *\|}{\|w, r, *\|\|*, r, w'\|}\right)$$
He uses this measure to define the similarity between two words as follows: Let $T(w)$ be the set of pairs (r, w') such that $\log\left(\frac{\|w,r,w'\|\|*,r,*\|}{\|w,r,*\|\|*,r,w'\|}\right)$

is positive. The similarity $sim(w_1, w_2)$ between two words w_1 and w_2 is:

$$\frac{\sum_{(r,w)\in T(w_1)\cap T(w_2)}(I(w_1,r,w)+I(w_2,r,w))}{\sum_{r,w\in T(w_1)} I(w_1,r,w)+\sum_{r,w\in T(w_2)} I(w_1,r,w)}$$

After evaluation of the resulting thesaurus and comparison with thesaurus and Wordnet, it was concluded that "our experiment surpasses previous experiments on automatic thesaurus construction in scale and (possibly) accuracy."

- Grefenstette(1994) [4] uses a shallow semantic parser to extract syntactic relationships in the form: (object,attribute,frequency) for different kinds of grammatical relationships. Grefenstette used the following function(using the notations described above) to measure the association(weight) of two words w and w':

  $$wgt(w,r,w') = log\left(\frac{(\|w,r,w'\|+1)}{(\|*,r,w'\|+1)}\right)$$

  Using the above weight function, similarity between two words w_i and w_j is calculated using the Jaccard Metric as follows:

  $$sim(w_i,w_j) = \frac{\sum_{(r,w')} min(wgt(w_i,r,w'),wgt(w_j,r,w'))}{\sum_{(r,w')} max(wgt(w_i,r,w'),wgt(w_j,r,w'))}$$

  In 2002, James Curan and Marc Moens [1] introduce several improvements over Grefenstette's such as introducing a new similarity metric and an approximation algorithm that bounds the time complexity of thesaurus extraction.

- In 2002, Pablo et al. [3]introduced elaborate attributes based on Grefenstette's set of attributes by modelling prepositional relationships and modifier-modified relationships explicitly. For example, in the prepositional phrases, "*permission to the company*" and "*authorization by the company*", the preposition "a" introduces quite a different syntactic dependency than the one introduced by the preposition "*by*". While the preposition "*to*" requires "*company*" to be the receiver within the action of giving authorization, the preposition "*by*" requires "*company*" to be the agent of the action.

  In addition to modelling relationships induced by prepositions, the modifier and modified relationships were also modelled explicitly. They used the jaccard similarity measure to calculate the similarity between words. It is concluded that their similarity measure which is based on fine-grained and elaborate syntactic contexts perform better than those based on poorly defined contexts.

1.3.3 Summary

A study of the previous literature on word similarity shows that there are two main approaches in defining contexts for word similarity models: windows-based approach and syntax-based approach. Different types of windows and syntax-based methods were examined in this chapter.

Both these approaches treat the neighboring words as a context for a given word. In this sense, the smaller-windows based method is quite similar to the syntax based approaches which consider the syntactic relationship to all the neighbouring words in a sentence or phrase.

However, there is a fundamental difference in these two methods. While the windows-based-approach treats the neighbouring words as a bag of words (with a few notable exceptions such as Schütze [16]), the syntax based method considers the relative positions which determine the syntactic relationships to the neighbouring words in the sentence.

Another difference between these methods is that many of the windows based methods use dimensionality reduction techniques such as singular value decomposition (SVD). SVD makes it possible to establish similarity between two words not only because they occur with the same set of contexts, but also because they occur with *similar* set of contexts. In the same way, two contexts may be similar as they may occur with a similar/same set of words. Such indirect relationships are called *latent relationships*[4].

Latent relationships are not realized in the syntax based approaches, that I have come across. Syntax based approaches establish similarity between two words as a function of their mutual information with the contexts they directly occur with. For instance, in Hindle's [6] method, described in the previous section, similarity between two nouns is computed using their respective mutual informations with respect to all the verbs.

Similarities that exists between the verbs is not captured in this equation. For intance, a noun n may have high mutual information with verb v_1 and low mutual information with verb v_2, as n cooccurs with relatively high frequency with v_1 than v_2. However, v_1 can be very similar to v_2, implying that n can potentially coocur with v_2, with a higher frequency than has been

[4]Chapter 3 describes this concept in greater detail

captured in the given corpus. This would increase the mutual information between n and v_2 from what was intially a low value. This *potential relationship* between n and v_2 is not taken into account in the computation of similarity between two nouns in most syntax-based methods .

This chapter has introduced word similarity modelling and reviewed previous literature in this area of research. The rest of the work describes word similarity modelling using a novel idea called "slot phrases" for modelling word similarity. It will be seen that slot phrase uses the relative positions between words within a context, like the syntax based approaches, and uses words-by-contexts matrix with SVD for modelling latent similarity, like the windows based approaches.

Chapter 2

Slot Phrases

2.1 Introduction

The definition of a context is the first and a vital step in constructing computational models of word similarity. In a general sense, the context for a given word can be considered as the totality of relationships of that word to the other words in the neighborhood. Various contexts have been proposed to capture this relationship of a word to its neighboring words. It was seen in the previous chapter that most of the research in defining contexts falls into one of the two approaches: windows based approach and syntax based approaches.

The windows based approach considers the context to be a bag of words within a specified neighborhood. This approach was seen to be further divided into two categories: larger windows based methods and smaller windows based methods.

Larger windows such as documents tend to convey topical information about a word. As topically similar words tend to occur within the same documents or occur in similar documents, the *larger-window-based method* tends to capture topical similarity. On the other hand, semantically similar words tend to share similar local set of local contexts. Therefore the *smaller-windows-based method* tends to capture more of semantic similarity.

Both the window based methods, with the exception of Schütze's approach, assume less prior linguistic knowledge and treat the neighboring words as a bag of words i.e. without considering the relative positions with other words in this window. As a result they provide no mechanism to explicitly model the linguistic relationships with the words in their neighbourhoods.

On the other hand the syntax-based approach uses prior linguistic knowledge to explicitly model syntactic relationships within this neighborhood. This approach is based on the assumption that semantically similar words tend to share the same syntactic relationships with other words in a neighborhood. Basing themselves on this assumption, syntax based methods use syntactic relationships to establish semantic similarity between words. However, syntax based methods do not use latent relationships to establish similarity between words, as discussed in the previous chapter.

The rest of this work aims at introducing and developing a novel idea called "slot phrases" as a context for word similarity models. The slot phrase-based approach shares certain similarities with the syntax-based approach and the windows-based approach.

This approach is similar to the syntax-based approach as it makes use of the relative positions between words and (implicitly) models syntactic relationships. On the other hand, it can be regarded as similar to the windows based approach as it involves the construction of a context-word matrix and singular value decomposition is performed on this matrix. The rest of this chapter is devoted to developing the idea of slot phrases.

2.1.1 Slot Phrases

Slot phrases are motivated by the idea that common patterns of words can act as good contexts for word similarity modelling. This idea is based on the assumption that similar words occur in similar *positions* with respect to common patterns of words. An example of a common pattern of words is "I went to the".This pattern usually occurs with places such as `restaurant,stadium` and `park` as illustrated below:

1. `I went to the restaurant`.
2. `I went to the stadium`.
3. `I went to the park`

It can be seen that the above pattern occurs as a context for various places. By using this pattern as a context for word similarity modelling, it is possible to establish similarity between words that refer to various places such as `restaurant,stadium` and `park`.

In order to capture this pattern and use it as a context, a slot phrase is constructed by substituting X for the words `office` and `school`:

I went to the X.

This example illustrates that slot phrases can be used to capture patterns of words which can act as contexts for word similarity models.

2.1.2 Some distinguishing features of slot phrases

- Slot phrases are intuitively appealing as natural contexts for word similarity modelling. They capture the intuition that similar words are *used* similarly by similar patterns of words.

- Slot phrases can be tuned to capture different kinds of relationships. For instance, a slot phrase with common words such as "I went to the X1 to X2" conveys that X1 may be places and X2 may be activities. Such *general slot phrases* can be used to model very general similarities such as syntactic similarities.

  On the other hand more specific slot phrases such as "I went to the restaurant to X" is a *specific slot phrase* which captures similarity between specific restaurant related words such as `eat, drink` and `dine`. In addition, slot phrases also provide a mechanism to model topical similarity by relating words in various slots within the same slot phrase.

- In contrast to the syntax based approach, slot phrases require very little prior linguistic knowledge. Common patterns of words, can be automatically constructed by identifying a spectrum of words from the most common to less common and then selectively substituting for these words with slots. Interestingly , domain specific slot phrases can also be constructed using the statistics of words in documents relating to specific domains.

- Despite using very little prior knowledge slot phrases may be expected to model various linguistic relationships between words. These relationships may not be limited to syntactic relationships used by the syntax based approach. This is because, slot phrases relate words directly instead of restricting the relationships based on abstract categories such as syntax.

- The slot phrase based approach aims at using statistical regularity to resolve ambiguities while modelling similarity. For instance a slot phrase can be syntactically ambiguous. For instance, "a X" is very general, and models similarity between nouns and adjectives at the same time. However the statistical regularity that arises from the presence of more specific slot phrases may be expected to overweigh such ambiguities.

These features of slot phrases make them look as an attractive alternative for a context. In my experiments, I have used slot phrases to construct word similarity models and compared them to some existing word similarity models. The next few sections describe the experiments using slot phrases.

2.2 Practical considerations in using slot phrases to define contexts

In the construction of computational models of word similarity using slot phrases, three main steps can be identified as summarized below:

1. Phrases are extracted from the text corpus according to some criterion.
2. Slot phrases are formed from these phrases by substituting for words (selected according to some criterion) by slots X. A data structure called the slot-word list, which establishes a correspondence between the words and the individual slots of each slot phrase is then constructed.
3. A *slot-by-word matrix* is constructed in which the cells record the number of times a word has occurred in a slot of a slot phrase.

The next two sections discuss the criterion for selecting phrases and for choosing the words to be replaced by the slots. The subsequent section (2.3) describes the steps involved in constructing the slot-word matrix starting from a text corpus.

2.2.1 Criteria for Identification of Phrases

The first step towards constructing slot phrases is to extract phrases from a text corpus according to some criteria. This criteria is naturally determined by their function, which is to serve as good contexts.

In order to function as good contexts, they should reflect the common patterns of words in texts. While it would be ideal to have a method which gives the most common patterns from a huge text corpus, I have taken a crude but fairly effective approach to identify these phrases.

I have used length and punctuation marks together, as criterion to select phrases. The intuition behind using length as criterion is that slot phrases which are too long or short do not act as good contexts. Long slot phrase being extremely specific, may rarely occur in a large enough corpus. On the other hand short slot phrases such as "a X", may not convey much information for word similarity models.

Punctuation marks(such as ,?!" etc) act as natural delimiters of syntactic units. Words within punctuation marks may not be the commonest patterns of words and for similar reasons, punctuation marks may not be the best way to find common patterns of words. However, I have provisionally used them for my experiments for the sake of keeping things simple.

2.2.2 Criteria for the selection of words to be replaced by slots

Once phrases have been identified, the next step is to convert them into slot phrases by substituting selected words with slots. The important question that arises in this regard is : *on what basis do we select words to be substituted with slots?*

- Substituting a slot for all but the most *common words* would result in slot phrases which are extremely common, therby establishing very general relationship between words. For instance, the slot phrase "The X" is a very common pattern and it conveys that "X" may be a noun or an adjective i.e. syntactic relationship. This example illustrates that general slot phrases such as these would tend to establish syntactic similarity.
- On the other hand, retaining some relatively *rarer words*, would give *less coverage*, while realizing more *specific relationships* between words[1].

The importance of this choice can be illustrated using the following example. Consider the following slot phrase: I went to the X1 to X2. This slot phrase is fairly general as it contains common words. This slot phrase can be interpreted as I went to the (some place) to (do some thing). This slot phrase conveys that X2 can be some *activity* usually performed in some place.

[1]Note that words cannot be classified into the twofold classification-rare and specific. There exits a spectrum from the most common and general to rare and specific words

This slot phrase, by itself gives no clues to distinguish between different types of generic activities. However, such finer distinctions between activities can be realized by retaining some of the relatively rarer words in the slot phrase. This point can be illustrated by an example. Consider the following sentences.

$$\text{I went to the }\textbf{restaurant}\text{ to}\begin{pmatrix}\textbf{eat}\\ \textbf{drink}\\ \textbf{relax}\end{pmatrix}$$

$$\text{I went to the }\textbf{stadium}\text{ to}\begin{pmatrix}\textbf{exercise}\\ \textbf{jog}\\ \textbf{run}\end{pmatrix}$$

Instead of substituting slot X1 for the words `restaurant` and `stadium`, if these were retained in the slot phrase "I went to the X1 to X2", we would have the *specific slot phrases*:
I went to the **restaurant** to X2
and
I went to the **stadium** to X2

Using the first slot phrase, it would be possible to characterize the words corresponding to X2 as restaurant related activities such as `eat,drink` or `relax`. Using the second, it would be possible to say that the words that can be substituted for X2 would be sports related activities such as `exercise`, `jog` or `run`.

This shows that by selecting the words to be retained in the phrases, according to whether those words are rare or common, it is possible to model specific and general similarity relationships between words. Speculatively speaking, it would be possible to model a *spectrum* of similarity relationships, from general to specific, by having a similar spectrum of slot phrases. The implementation of this idea is dealt with in the next section.

2.3 Practical steps towards constructing contexts using slot phrases

The following are the sequence of steps I have taken in constructing a slot-by-word matrix, starting from a text corpus.

1. **Text Corpus**:A text corpus(1 million sentences) is assembled by putting together, a collection of all children's literature available as e-texts from the Gutenburgh Project. The official website for this project is http://www.gutenberg.org/. The corpus is constructed by collecting all the children's stories in the Children's literature section of the Gutenberg corpus. This section has 10 pages each of which have a list of stories. The web page of the first page is: http://www.gutenberg.org/catalog/world/results?locc=PZ&pageno=1. The rest of the 9 pages can be accessed by substituting the page number in 'pageno=' of this URL. The corpus has around 1 million lines.

2. **Extraction of Phrases:** Phrases with atleast 3 closed class words and separated by punctuation marks are then extracted.

3. **Generating Slot Phrases:** Slot Phrases are derived from these phrases by substituting for words with slots(X). A data structure called the *slot-by-word table* is created to record the correspondence(can call it the *slot-word entries*) of the slots with the substituted words.

   However, the derivation of slot phrases is carried out in two stages. Two collections of slot-phrases were identified in this studies: general and specific slot phrases. First, general slot phrases are derived and the corresponding slot-by-word table is formed. Next, the entries in this table are used to generate specific slot phrases and the specific slot-word entries are added to this table.

   (a) General Slot Phrases: A list of the thousand high frequency words of the corpus are first collected. All the words except those in this list are then substituted with a slot(X) and a table which records the slot phrases and the words that were present in them are recorded in a table.
   For example, consider the following Sentences and their equivalent slot Phrase

   I **ironed** all my **clothes** for the **trip**
   I **packed** all my **things** for the **journey**
   I **X** all my **X** for the **X**

   This Slot Phrase and the corresponding slot-word entries are added to the slot phrase-word table in the following format[2].

[2]I have left out slot phrases which have two content words together for the moment.

I X all my X for the X - `ironed:clothes:trip + packed:things:journey`

According to this format, the words in different slots of the same slot phrase are separated by ":" .This forms a set of words such as `ironed:clothes:trip`. These sets are then listed and separated by "+".

It is important to note that only those general slot phrases in which atleast 10 words have occurred in each of the slots have been recorded in the slot word table. This decision can be justified on the basis of a preliminary study, according to which 98% of the slot phrases have less than 5 words in their slots[3]. However, it was observed that the 2% of the slot phrases covered more than 90% of the words. Therefore, only this 2% of the slot phrases have been used for forming the slot-word table.

(b) <u>Specific Slot Phrases:</u> Specific slot-word entries are generated using entries in the slot-word table formed in the previous step.

   i. General slot phrases having more than 1 slot are first considered. The following is an example of such a slot phrase.

      I went to the X1 to X2:`stadium:jog+club:jog+park:jog+stadium:exercise....`

   ii. Words that occur in atleast 3 general slot phrases are identified. In the above example `jog` appears three times. Note that the word `jog` corresponds to the slot X2 of the above slot phrase.

   iii. A *specific slot phrase* is constructed by substituting the frequent words(`jog` in this case) in its corresponding slot. In the above example, the *specific slot phrase* would be I went to the X1 to **jog**

   iv. The words that appear with this frequent word (`Jog`) such as `club, park` and `stadium` are then extracted and a slot-word entry is formed for the specific slot phrase. In this example, the specific slot-word entry would be

For instance the following phrase: `I neatly packed my the shirts for the trip` which corresponds to the slot phrase I X X all my X for the X would not be left out in the data structure

[3]This was due to the use of punctuation marks which often gave rise to long slot phrases, which are extemely rare

I went to the X1 to **jog**:- `club:park:stadium`.

Repeating the above steps for every entry of the general-slot-word-table and adding the specific-slot-word mappings to this table, the final slot word table is created.

It can be seen that these specific slot phrases serve to establish further similarity between words. In this example the slot phrase I went to the X1 to Jog establishes *further similarity* between the words `club, park` and `stadium`[4]

(c) **Slot-by-Word Matrix** From the slot-by-word table , a slot-word matrix, is obtained. $Cell_{i,j}$ of this matrix contains the frequency of the $word_j$ in $slot_i$. This matrix is obtained in the following two steps:

i. Slot Word List First, a slot-word list is constructed. This is a list of words in each slot(of each of the slot phrases). However, instead of a list of words themselves, the indexes[5] of the slots and the words are used to represent the correspondence. For instance, in the above example, if ironed and packed have indexes w_i and w_j and the index of the first slot of the slot phrase I **X** all my X for the X be x_m, this correspondence would be represented in the list in the following manner[6]:

x_m: w_i, w_j...

ii. Slot-Word Matrix In the second step, a sparse slot-by-word matrix is constructed using the above list. Considering the same example, the corresponding slot word matrix would have the following entries:

....
x_m w_i $n_{m,i}$
x_m w_j $n_{m,j}$
....

[4]Note that the general slot phrase "I went to the X1 to X2 " does not specifically make `club, park` and `stadium` similar.

[5]This index is obtained in the previous step

[6]As the same SP can have many slots, the j'th slot of the i'th SP is indexed as $j \cdot maxsize + i$, where $maxsize$ is the number of SPs in the SP list.This indexing scheme has been chosen for convenience.

This slot matrix can be considered a vectorial representation of words capturing both coverage and specificity.

2.4 Low Dimensional Representation using SVD

The above steps results in a slot-word matrix- a high dimensional representation of words. To obtain a more compact low dimensional representation, singular value decomposition is performed on this matrix. This results in the word similarity model. It is expected that this low dimensional representation would capture latent relationships between words.

Investigations of this low dimensional representation is discussed in chapter 4, after an introduction to the theory behind the machine learning methods used in this work.

Chapter 3

Introduction to machine learning methods used in this work

3.1 Introduction

This chapter discusses the four main machine learning methods that I have used in my experiments with slot phrases. These are singular value decomposition(SVD), K-means clustering, support vector machines (SVM) and kernel canonical correlation analysis (KCCA). While SVD is used to obtain a low dimensional representation of words from the slot word matrix, the other three methods are largely used as exploratory tools to understand this space and compare them with others.

The following section discusses singular value decomposition with a synthetic example to gain an intuition on its effect on the much larger slot word matrix. Clustering is discussed in section 2. Sections 3 and 4 discuss support vector machines and KCCA respectively.

3.2 Singular Value Decomposition

3.2.1 Introduction to Singular Value Decomposition

Singular Value Decomposition is a technique used to obtain a low dimensional representation which may capture the latent structure of a high dimensional data set. In the context of word similarity models, they are used to obtain a low dimensional word similarity model from a high dimensional representation of words as a *slot-word matrix* introduced in the previous chapter.

The slot-word matrix represents a word in terms of its occurrence in each of the slots. According to this representation, two words are more similar, the more they tend to share the *same set* of slots. However, two words may be similar because they tend to share many *similar slots* and not necessarily the same slots. Likewise, two slots may be very similar because they share many similar words and/or the same set of words.

Such indirect relationships are called *latent relationships* and are not captured in the slot-word matrix. SVD creates a lower dimensional space in which it captures latent relationships between words. In the context of this work, we will term this space as the *word space.* It creates the word space by first decomposing the high dimensional matrix A into U and V in the following way:

$$A = USV'$$

Here, U and V are orthogonal matrices, the eigenvectors of $A'A$ and AA' respectively. S is the diagonal matrix whose diagonal elements are non negative square roots of the eigenvalues of $A'A$ and AA'.

Since the rows of A' are word vectors, $A'A$ becomes the *word similarity matrix.* The eigen vectors of $A'A$ - i.e. the rows of U are might be termed as *eigenwords.* Similarly, the space of V can be called the *eigencontexts.*

Projection of the word vectors on the eigenword vectors would then result in what can be called the *word space.* It is this low dimensional word space that is expected to capture better similarity relationships between words.

In the next section, it is proved that every matrix admits of SVD. The material in this section follows the discussion in John Shawe Taylor's book on Kernels [20] and the discussion of SVD by Hourigan et. al. [18] Section 3.3 illustrates how SVD reveals the latent relationship between words with a demonstration example.

3.2.2 Mathematical Proof of Singular Value Decomposition

The following theorem proves that every matrix admits of SVD [20]

Theorem 1. *Every matrix admits of singular value decomposition.*

Proof. To prove the above theorem, it will be shown below that every matrix A can be decomposed into the singular matrices U and V. Let A be an $(m \times n)$ matrix.

Consider the correlation matrix $C = A'A$ and the Gram matrix $K = AA'$, which are both positive semidefinite symmetric matrices and therefore can be diagonalized.

It is known that for a symmetric matrix, the following is true.

1. The eigen values are real and non-negative.
2. The eigen vectors will be orthogonal.

Taking the eigen value decomposition of We have :

$$C = A'A = U\Lambda_c U' \tag{3.1}$$

$$K = AA' = V\Lambda_k V' \tag{3.2}$$

Let v be an eigen vector of K. Consider $CA'v$. Substituting for C from equation 3.1, we have,

$$CA'v = (A'A)A'v = A'(AA')v \tag{3.3}$$

Substituting for AA' with K(equation 3.2) into equation 3.3, we have

$$CA'v = A'Kv \tag{3.4}$$

Since v is a eigenvector of K (equation 3.2), substituting $Kv = \lambda_k v$ into the above equation gives:

$$CA'v = \lambda_k A'v \tag{3.5}$$

This implies that λ_k and $A'v$ is an eigenvalue-eigenvector pair for C. In a similar way, we can take an eigenvector-eigenvalue pair u, λ_c of C, and transform it to a eigenvector-eigenvalue pair Au, λ_c of K. This means that the set of eigenvalues for both C and k are the same. we therefore have $\lambda_k = \lambda_c = \lambda$, where λ is the common set of eigen vectors.

The norm of $A'v$ is given by :

$$\|A'v\|^2 = v'AA'v = \lambda \tag{3.6}$$

This implies that the normalized eigenvector u of C is $u = \lambda^{\frac{-}{1}}2A'v$
By symmetry, we have the normalized eigenvector v of K is $v = \lambda^{-\frac{1}{2}}Au$.

Taking r as the rank of K, we have

$$U_r = A'V_r\mathbf{\Lambda}^{-}\frac{1}{2} \tag{3.7}$$

It is assumed here that r non-zero eigen values of K and C appear in descending order. By the symmetry of C and K, these are the only non-zero eigen values of C, since we can transform any eigenvector-eigenvalue pair u, λ of C to an eigenvector-eigenvalue pair Au, λ of K. It follows then:

$$r = rank(K) = rank(C)$$

It can now be proved that r is also the rank of A.

Lemma 1. *Let the rank of A be equal to r. Then r is also equal to rank($A'A$), which is also equal to the number of non-zero eigenvalues.*

Proof. Let $x \in N(A)$, where $N(A)$ is the nullity of A. Then, it follows that $Ax = \mathbf{0}$. Multiplying by A', we have

$$A'Ax = A'\mathbf{0}$$

$$A'Ax = \mathbf{0}$$

This implies that $x \in N(A'A)$. Therefore

$$N(A) \subset N(A'A) \tag{3.8}$$

Suppose $x \in N(A'A)$, then we know that $A'Ax = \mathbf{0}$. Consider $\|Ax\|^2$. We have

$$\|Ax\|^2 = x'A'Ax = x'(A'Ax)$$

Since $A'Ax = \mathbf{0}$,

$$\|Ax\|^2 = 0$$

or

$$\|Ax\| = 0$$

$$x'A'Ax = \mathbf{0}$$

$$(Ax)'Ax = 0$$

$$\|Ax\|^2 = 0$$

$$\|Ax\| = 0$$

This implies that $x \in N(A)$. Since $x \in A'A$, we have

$$N(A'A) \subset N(A) \tag{3.9}$$

Combining equation 3.8 and 3.9, we have

$$N(A'A) = N(A) \tag{3.10}$$

Let A have dimension $m \times n$. $A'A$ has dimension $n \times n$. It follows from 3.10 that the nullity of $A'A =$ nullity of A and is equal to k, implying that

$$\text{rank } (A'A) = n - k \tag{3.11}$$

$$\text{rank } (A) = n - k \tag{3.12}$$

Combining these two results, we have proved the theorem:

$$\text{rank of } A'A = \text{ rank of } A = r$$

$\square$

Rewriting equation 3.7, we have

$$U_r = A'V_r\mathbf{\Lambda}^{-\frac{1}{2}} \tag{3.13}$$

This implies that

$$A' = U_r\mathbf{\Lambda}^{\frac{1}{2}}V_r \tag{3.14}$$

By extending U_r, V_r and $\Lambda^{\frac{1}{2}}$ to $(m \times n)$ matrices $U = [U_r, U_{r+1..n}]$, $V = [V_r, V_{r+1..m}]$ and S, whose additional entries are all zero, we have

$$USV' = [U_r, U_{r+1..n}] \begin{pmatrix} \Lambda^{\frac{1}{2}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} [V_r, V_{r+1..m}]$$

,

$$= [U_r, U_{r+1..n}] \begin{pmatrix} \Lambda^{\frac{1}{2}}V_r \\ \mathbf{0} \end{pmatrix}$$

$$= U_r\Lambda^{\frac{1}{2}}V_r + \mathbf{0}$$

$$USV' = U_r\Lambda^{\frac{1}{2}}V_r \tag{3.15}$$

Combining the results of equation 3.14 and 3.15, we have the singular value decomposition of A:

$$A' = USV'$$

where S is a $m \times n$ matrix with all entries zero except the leading diagonal which has entries $s_i = \sqrt{(}\lambda_i)$, where the r s_is are in ascending order. This completes the proof. $\square$

Lemma 2. *The matrix $A_k = U_k S_k V_k'$ formed by the cross product of the first k columns of $U(U_k)$ and $V'(V_k')$ and the first k diagonal elements(S_k) will be the* least squares-best fit *differing from the original matrix A by the k'th eigen value.*

It will be seen that this lower dimensional space A_k is a truer representation of the structure of the space.

3.2.3 Synthetic Experiments to Illustrate the working of SVD

The previous section introduced SVD as a technique to obtain a low dimensional representation of a high dimensional space, which may capture the latent structure of this space. In the context of word similarity models using slot phrases, SVD of the slot-word matrix would result in a low dimensional space. The natural question would be: can we expect this space to reveal the latent relationships between words?

In order to gain an intuition of what SVD does, I have constructed a synthetic example. I have first created two categories of similar words:words similar to *observed, seen* etc and words similar to *prefer, like* etc. I have found slot phrases that use these words and used SVD on the slot word matrix. As, it will be seen, this slot matrix does not represent the true similarity relationship between the words. I have deliberately chosen the slot phrases so that word similarity relationships are not expressed directly but *latently.* The experiment tries to examine whether SVD has brought about this latent relationships. The experiment is described in greater detail below.

1. Nine slot phrases with 12 words are considered. The words and the slot phrases are constructed so that there are two semantic categories of words. The first category is similar to *observe, seen etc* and the second to *like, prefer* etc. 5 slot phrases correspond to the first type and 4 to the second type.These listed below.

2. A slot-word matrix is formed from these slot phrases and words as shown in Table 1.

3. SVD is performed on this matrix to obtain U, S and V from A.

4. The first two eigen values along the diagonal of S is found to be much higher than the rest.Therefore U_2 and V_2 are constructed from U and V by taking only the first two columns and rows respectively. S_2 is constructed by taking the first two row and columns of S.

5. The matrix $A_2 = U_2 S_2 V_2$ is formed as shown in table 2.

It can be shown by a few observations that A_2, which is an approximation to A (according to the lemma in previous section) has revealed a truer relationship between the words.

Category 1

- SP1: I have always X that :observed, noticed, noted
- SP2: It can be X that :**inferred**, seen, perceived,noted, shown, proved.
- SP3:The mark was X by every one: seen,noticed,perceived,commented upon
- SP4:As he correctly X: perceived 2,observed,commented upon
- SP5:It could be X by anyone: seen,shown, proved

Category 2

- SP6:I always X: prefer
- SP7:He some how X: likes,prefers
- SP8:I don't X: like,appreciate,prefer,
- SP9:Some people X: like,appreciate,**inferred**.

Slot phrases and the words they take.

Conclusions Comparing the original matrix A with A_2, the following conclusions can be drawn about how A_2 has captured true relationships between words.

1. Consider the word *infer*. It occurs in SP2 and SP9 in the original matrix A and has the value 1. However this word belongs to the first category and therefore belongs more to SP2 than to SP9. This fact has been revealed in the low dimensional approximation A_2 where its value in SP9 has decreased to .42. This means that there is a smaller chance of the word *infer* occurring in SP9. In other words the lower dimensional representation has captured the latent relationship between the slot-SP9 and the word-*infer*

SP	$SP1$	$SP2$	$SP3$	$SP4$	$SP5$	$SP6$	$SP7$	$SP8$	$SP9$
Observed	1	0	0	1	0	0	0	0	0
noticed	1	0	1	0	0	0	0	0	0
noted	1	1	0	0	0	0	0	0	0
seen	0	1	1	0	1	0	0	0	0
perceived	0	1	1	2	0	0	0	0	0
shown	0	1	0	0	1	0	0	0	0
proved	0	1	0	0	1	0	0	0	0
C.U	0	0	1	1	0	0	0	0	0
inferred	0	1	0	0	0	0	0	0	1
prefer	0	0	0	0	0	1	1	1	**0**
like	0	0	0	0	0	0	1	1	**1**
appreciate	0	0	0	0	0	0	0	1	1

Table 3.1: The Original Data Matrix A. Note: (a) C.U=Commented upon (b)The morphological variations such as inferred (of infer) are considered the same as their root words in forming the slot-word matrix. (c) SP-refers to slot phrases.

SP	$SP1$	$SP2$	$SP3$	$SP4$	$SP5$	$SP6$	$SP7$	$SP8$	$SP9$
observed	0.1600	0.4000	0.3700	0.4600	0.1800	-0.0500	-0.1200	-0.16	-0.09
noticed	0.1400	0.3800	0.3300	0.4000	0.1700	-0.0300	-0.0700	-0.1000	-0.04
noted	0.1500	0.5100	0.3600	0.4100	0.2400	0.0200	0.0500	0.0800	0.12
seen	0.2600	0.8400	0.5900	0.6900	0.3900	0.0300	0.0800	0.12	0.19
perceived	0.4500	1.2300	1.0400	1.2500	0.5500	-0.0800	-0.1700	-0.22	-0.06
shown	0.1600	0.6000	0.3800	0.4200	0.2800	0.0500	0.1300	0.1900	0.22
proved	0.1600	0.6000	0.3800	0.4200	0.2800	0.0500	0.1300	0.1900	0.22
C.U	0.2200	0.5500	0.5100	0.6200	0.2400	-0.0700	-0.1500	-0.20	-0.11
inferred	0.1000	0.5400	0.2300	0.2200	0.2700	0.1300	0.3100	0.4400	0.42
prefer	-0.0700	0.2300	-0.1500	-0.2700	0.1500	0.2400	0.5500	0.77	**0.66**
like	-0.0700	0.3500	-0.1400	-0.2900	0.2100	0.3000	0.6900	0.98	**0.85**
appreciate	-0.0500	0.2600	-0.1000	-0.2100	0.1500	0.2200	0.50	0.7100	0.61

Table 3.2: The reconstructed Matrix-$A_2 = U_2 S_2 V_2'$ of A Using 2 eigen values and vectors

2. The word *prefer* has not occurred in SP9 and has the value 0 in the original matrix A. However, the word prefer belongs to category 2 and should occur with a greater chance in the slot of SP9. This is reflected in the matrix A_2, where the word *prefer* has a greater value-.66 in SP9. Thus, SVD has been able to reveal the latent relationship between a word and a slot.

3. Consider the words *observed* and *seen*. From the original data matrix, it can be seen that they share no slot phrases and their correlations are extremely low. However, in the matrix A_2, their row values are much more similar and so their correlation is much higher. This is due to the fact that these two words are not directly related, rather they are *latently related*. SVD has brought out this latent relationship between two similar words, whose similarity has not been reflected in the slot-word matrix.

Discussion: The conclusions have shown that the matrix A_2 is a truer representation of the relationship between words.The next chapter describes the use of SVD in constructing the word space. The slot-word matrix obtained from a text corpus is a high dimensional matrix. SVD on this matrix produces a low dimensional representation of words. The expectation is that this low dimensional matrix would be a truer representation of some similarity relationships between words than the slot word matrix. The synthetic example described above can be considered a strong motivation in order to have similar expectation about using SVD on the much bigger slot-word matrix.

3.3 Clustering

Clustering refers to a class of algorithms which try to partition a data set in a vector space into clusters. When the points are not naturally present as clean clusters, these algorithms usually partition the vector space into regions in which points are densely populated.

While clustering can be used to automatically group words into classes, I have used clustering as a tool for exploratory analysis of the word space. It will be seen in the next chapter that there are hardly any clean clusters that can be expected in the complex word space. Therefore, I have used clustering to understand the distribution of words in the word space so as to answer the question: “Does closeness imply similarity in the word space?”.

There are many algorithms which have been used for clustering. The popular algorithms are the k-means, soft k-means, hierarchical clustering algorithms, spectral clustering and kernel-k-means clustering. I have implemented several of these algorithms and tried them in pilot studies, but have simply used k-means algorithm in these experiments reported. The following paragraph describes this algorithm.

3.3.1 k-means Clustering

The k-means Clustering algorithm is one of the simplest clustering algorithms that aims to partition the data into k-dense regions, through several iterations. The k-means algorithms proceeds iteratively in two stages.

1. k points are generated at random in the region in which the clusters lie (usually this is achieved by choosing k points randomly from the data set). The distances of all the points to these k points are calculated and to each data point, the nearest of the k points is assigned. This results in partition of the data into classes(atmost k) with each of the k points representing a class.

2. For each of the classes formed in the first step the mean is taken to represent the cluster center of that class.

3. Steps 1 and 2 are repeated till the data is partitioned into k clusters.

The k-means is simple to implement and the results can be viewed on the screen using principle component analysis (PCA), which produces a low dimensional representation of the data. I have used the k-means algorithm and PCA in my experiments to analyze the word space and view them in two dimensions to see if there are some obvious clusters.

3.4 The Support vector machine

3.4.1 Introduction

The Support Vector Machine (SVM) is a supervised learning formulation introduced by Vapnik [23] for pattern recognition. This formulation embodies the structural risk minimization principle introduced by Vapnik in his work on statistical learning theory where the performance of the SVM binary classifier is determined by controlling its capacity and minimizing the training error on the data set.

The simplest case for SVM classification problem is the *binary classification task*, where the positive and negative examples are *linearly separable* by a hyperplane. This is illustrated in fig 1 on the next page. Among all the possible hyperplanes which separate the positive and negative training points, SVM selects the one where the distance of the hyperplane from the closest positive and negative examples is as large as possible. This hyperplane is called the *optimal hyperplane* as it is expected to have minimum error on the test set, or best generalization.

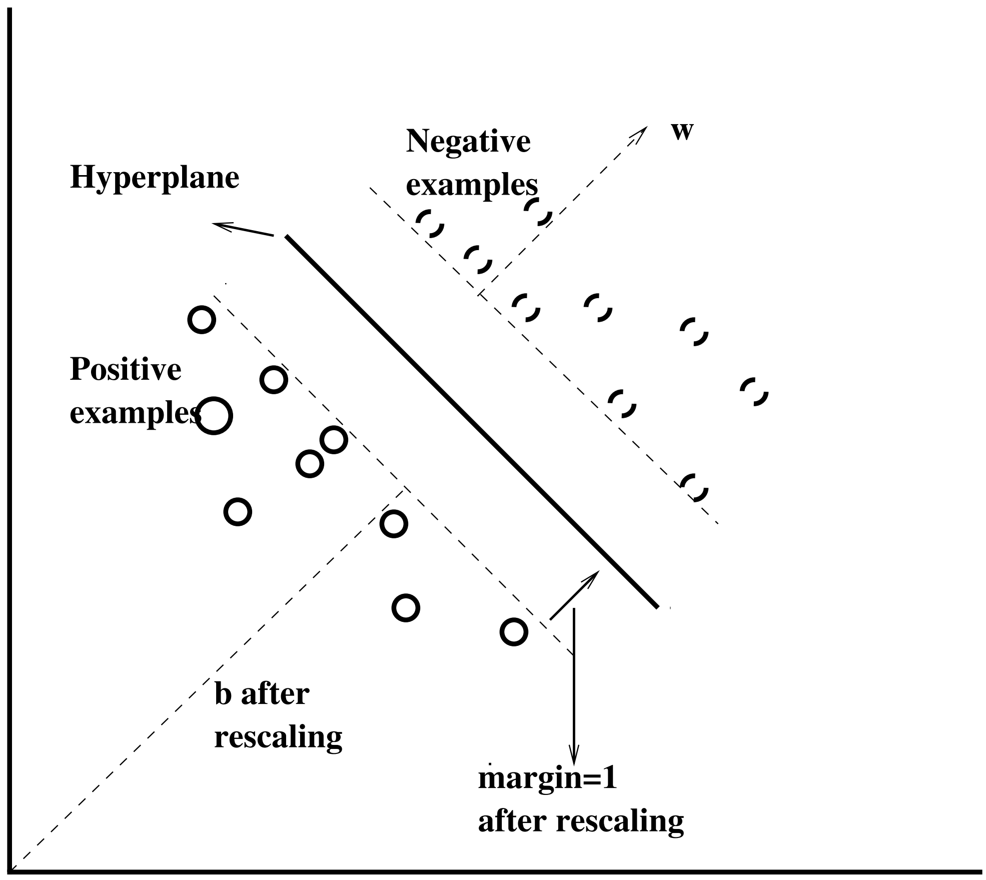

Figure 3.1: SVM classification for linearly separable examples.

The following section describes the mathematical formulation of the SVM problem, leading to the primal form. Section 1.3 shows that the primal and the dual formulation have the same optimal values.Section 1.4 describes the dual formulation of the SVM problem. The final section describes how kernels can be incorporated in the dual form. In these sections, I have followed the discussion of SVM in Scholkopf et.al. [13] and John Shawe Taylor and Nello Cristianini's book on SVM [19].

3.4.2 The classification problem and primal formulation of the SVM

The standard classification problem can be formulated as follows: Let $\mathbf{x}_i$ for $i =$ 1 to l be the training vectors divided into positive and negative examples which we assume, can be linearly separated by a hyperplane. Let y_i be their labels, where $y_i = +1$ for positive examples and $y_i = -1$ for negative examples. If $\mathbf{w}$ is the normal vector of the separating hyperplane and b is the distance from the origin to the hyperplane. The equation of the separating hyperplane is

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

and the classification function f is

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b \tag{3.16}$$

and that of the classification where $f(\mathbf{x}) > 0$ for positive examples and $f(\mathbf{x}) < 0$ for negative examples.

This means that for all examples $\mathbf{x_i}$, $i = 1..l$,

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) > 0$$

$\mathbf{w}$ and b can be rescaled so that we have for all i,

$$y_i((\mathbf{w} \cdot \mathbf{x_i}) + b) \geq 1 \tag{3.17}$$

This implies that the points $\mathbf{x_i}$, closest to the hyperplane satisfy,

$$\|y_i((\mathbf{w} \cdot \mathbf{x_i} + b)\| = 1 \tag{3.18}$$

The margin can be calculated by consider two closest points $\mathbf{x}_1$ and $\mathbf{x}_2$ on either side of the plane. According to the previous equation 10,

$$(\mathbf{w} \cdot \mathbf{x}_1) + b = +1$$

$$(\mathbf{w} \cdot \mathbf{x}_2) + b = -1$$

so that

$$(\mathbf{w} \cdot (\mathbf{x}_1 - \mathbf{x}_2)) = 2$$

thus the projection of $\mathbf{x}_1 - \mathbf{x}_2$ along the direction of the normal vector $\mathbf{w}$ is

$$\frac{\mathbf{w}}{\|\mathbf{w}\|}(\mathbf{x}_1 - \mathbf{x}_2) = \frac{2}{\|\mathbf{w}\|}$$

Thus, the margin for the *canonical hyperplane* is $\frac{2}{\|\mathbf{w}\|}$. Maximization of this margin results in the optimum hyperplane that separates the positive and negative points.

Therefore the SVM problem can be formulated as follows:

$$\text{minimize } \frac{1}{2}\|\mathbf{w}\|^2 \text{ subject to } y_i((\mathbf{w} \cdot \mathbf{x_i}) + b) \geq 1, i = 1....m \tag{3.19}$$

This form of the optimization problem is known as the *primal form.* The next section shows that the corresponding dual has the same optimal values as the primal.

3.4.3 Same optimal values for the primal and the dual form

Often for many primal problems, such as equation 11 there exists the equivalent dual problem, which is often easier to solve. Under certain conditions, the primal and the dual have the same optimum values, implying that the problem can be solved in the equivalent dual form. The following passages show that the primal and the dual problem share the same optimum values.

The above formulation has its corresponding dual:

$$\max_{\alpha \geq 0}(\min_{\mathbf{w},b} L(\mathbf{w}, b, \alpha))$$

where

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}^2\| - \sum_{i=1}^{l} \alpha_i(y_i(\mathbf{w} \cdot \mathbf{x_i} + b) - 1) \tag{3.20}$$

This means that the Lagrangian L has to be *minimized* with respect to the primal variables $\mathbf{w}$ and b and *maximized* with respect to the dual variables α_i.

Let $(\overline{\mathbf{w}}, \overline{b})$ be the optimal solution of the *primal* with the optimal objective value $\gamma = \frac{1}{2}\|\overline{\mathbf{w}}\|^2$.

Thus no $(\mathbf{w}, b)$ satisfies

$$\frac{1}{2}\|\mathbf{w}\|^2 < \gamma \text{ and } y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1,\ i = 1....m \tag{3.21}$$

Therefore, there is a $\overline{\alpha} \geq 0$ such that all $(\mathbf{w}, b)$ satisfy:

$$\frac{1}{2}\|\mathbf{w}\|^2 - \gamma - \sum_{i=1}^{m} \overline{\alpha_i}(y_i(\mathbf{w} \cdot \mathbf{x_i} + b) - 1) \geq 0 \tag{3.22}$$

The proof for this can be found in Bazaraa et.al [12].

By substituting for L from equation 12, we can rewrite this as:

$$\max_{\alpha \geq 0}(\min_{\mathbf{w},b} L(\mathbf{w}, b, \alpha)) \geq \gamma \tag{3.23}$$

On the other hand,since $(\overline{\mathbf{w}}, \overline{b})$ are the optimal solution for the primal, for any α:

$$\min_{\mathbf{w},b} L(\mathbf{w}, b, \alpha) \leq L(\overline{\mathbf{w}}, \overline{b}, \alpha) \tag{3.24}$$

Using this inequality, we have,

$$\max_{\alpha \geq 0}(\min_{\mathbf{w},b} L(\mathbf{w}, b, \alpha)) \leq \max_{\alpha \geq 0} L(\overline{\mathbf{w}}, \overline{b}, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 = \gamma \tag{3.25}$$

Rewriting the inequalities in equation 15 and 17 together,we have:

$$\max_{\alpha \geq 0}(\min_{\mathbf{w},b} L(\mathbf{w}, b, \alpha)) \geq \gamma \tag{3.26}$$

$$\max_{\alpha \geq 0}(\min_{\mathbf{w},b} L(\mathbf{w}, b, \alpha)) \leq \gamma \tag{3.27}$$

This gives rise to the equality:

$$\max_{\alpha \geq 0}(\min_{\mathbf{w},b} L(\mathbf{w}, b, \alpha)) = \gamma \tag{3.28}$$

This equation means that the optimal value of the dual is the γ, the same as that of the primal. This implies that instead of solving the primal, we can try to solve the equivalent dual form. Before the dual form is solve the following condition is derived as it would be useful later:

Substituting $(\overline{\mathbf{w}}, \overline{b})$ into equation 3.14, we have

$$\sum_{i=1}^{m} \alpha_i(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1) \geq 0 \tag{3.29}$$

Since, $\alpha_i \geq 0$ and $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0$ for all i, we get

$$\alpha_i(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1) = 0 \text{ for all } i = 1..l \tag{3.30}$$

This is called the *complementarity condition.* According to this condition, either $\alpha_i = 0$ or $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 = 0$. The significance of the complementarity condition will be explained in the next section after the derivation of the dual form for support vector machines.

3.4.4 The SVM dual form

Rewriting the dual, we have

$$\max_{\alpha \geq 0}(\min_{\mathbf{w},b} L(\mathbf{w}, b, \alpha))$$

where

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}^2\| - \sum_{i=1}^{l} \alpha_i(y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1) \tag{3.31}$$

Since the optimum values of $\mathbf{w}$ and b are to be found, the partial differentials with respect to $\mathbf{w}$ and b are taken and equated to 0,

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = 0, \frac{\partial (L(\mathbf{w}, b, \alpha)}{\partial b} = 0 \tag{3.32}$$

This leads to

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{3.33}$$

and

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \tag{3.34}$$

The equation 3.26 expresses $\mathbf{w}$ in terms of α_i. Therefore $\mathbf{w}$ can be substituted into the expression for L in equation 12, so that the dual form is simplified to finding multipliers α_i which maximizes:

$$\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j \tag{3.35}$$

subject to

$$\alpha_i \geq 0 \text{ for } i = 1...m \text{ and } \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{3.36}$$

This is usually referred to as the SVM dual form. It was seen that the hyperplane decision function for the primal form was

$$f(\mathbf{x}) = \sum_{i=1}^{l} (\mathbf{w} \cdot \mathbf{x}_i + b) \tag{3.37}$$

Using the dual formulation, the hyperplane decision function becomes:

$$f(\mathbf{x}) = \sum_{i=1}^{l} y_i \alpha_i (\mathbf{x} \cdot \mathbf{x}_i) + b \tag{3.38}$$

Using the complementarity condition (equation 22) it can be inferred that since,$((y_i(\mathbf{w} \cdot \mathbf{x}_i) + b) - 1) > 0$ for all other points other than the points that lie on the margin, α_i should be zero for all examples other than the support vectors which lie in the margin ie. the *support vectors.* Intuitively speaking these points support the hyperplane, if we interpret this problem as being analogous to a physical problem where each of the examples are exerting a force in the direction away from the hyperplane [13].

The conditions for equilibrium would be achieved when the forces and torques respectively cancel each other. Looking at the problem this way it can be seen how equations

$$\sum_{i=1}^{l} (\alpha_i y_i) = 0 \tag{3.39}$$

and

$$\mathbf{w} = \sum_{i=1}^{l} (\alpha_i y_i \mathbf{x}_i) \tag{3.40}$$

express this condition. As the examples that are not support vectors are irrelevant, it can be seen that the α_i's give the respective forces for these vectors so that the optimal hyperplane can be obtained.

This form of the hyperplane is advantageous in two ways. Firstly, once the data is trained we would have to consider only the support vectors in order to calculate the expression for the test data. Secondly, as the dual form expresses the result in the form of dot products we can use kernels compute the classification function without explicitly calculating the mappings and then computing the dot product.

3.4.5 Incorporating Kernels in the Dual Form

The SVM problem and its solution as described in the previous section assumed that the examples are linearly separable. But this may not be the case. However, it might be possible to separate the data with non linear surfaces such as ellipsoids. But, the SVM is a learning algorithm which constructs a hyperplane to separate linearly separable data.

Instead of trying to construct a non-linear surface, this problem is solved by mapping the data points to a space, where the points are linearly separable by a hyperplane. Usually, it is the case that when linearly non-separable data is cast into a high dimensional space, it becomes linearly separable. Thus, the data can be mapped into another space by a function $\phi(\mathbf{x})$, where it becomes linearly separable.

However, handling very large feature vectors would become a computational burden. But the dual form suggests an easy way out. It can be observed from the dual form and the formulation of the hyperplane (equation 27,28) that the dot product $\mathbf{x} \cdot \mathbf{x}_i$ figures in both these expressions. In the high dimensional space, the equivalent dot product would be $\phi(\mathbf{x}) \cdot \phi(\mathbf{x}_i)$.

This computationally costly dot product can be avoided by the *kernel trick*, which uses a formula given by

$$\mathbf{K}(\mathbf{x}, \mathbf{x}_i) = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}_i)$$

to implicitly compute the scalar product for very large feature vectors.

The SVM classification problem can be reformulated in terms of Kernels as follows:

$$\sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \mathbf{K}(\mathbf{x_i}, \mathbf{x_j}) \tag{3.41}$$

$$\text{subject to } \alpha_i \geq 0 \text{ for } i = 1...m \text{ and } \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{3.42}$$

The classification function resulting from the solution to the above problem is:

$$f(\mathbf{x}) = \sum_{i=1}^{l} y_i \alpha_i \mathbf{K}(\mathbf{x}, \mathbf{x}_i) + b \tag{3.43}$$

Choosing, adapting and designing kernels for particular applications is often considered a matter of skill and experience. However, certain features of some standard kernels are quite well known. Some standard kernels are polynomial kernels, gaussian kernels etc.

The Gaussian kernel is quite popular and is known for forming connected island like structures that separate examples. The Gaussian kernel is:

$$\mathbf{K}(\mathbf{x_a}, \mathbf{x_b}) = \mathbf{exp}\left(\frac{-\|\mathbf{x_a} - \mathbf{x_b}\|^{\mathbf{2}}}{\mathbf{2}\sigma^{\mathbf{2}}}\right)$$

where σ can be tuned. I have used SVM with Gaussian kernels to capture syntactic categories within a region in the word space. This is described in greater detail in chapter 4 on experiments.

3.5 Canonical Correlation Analysis

Canonical correlation analysis (CCA) can be seen as way of comparing two views of the same sets of objects represented in vector spaces by finding linear relationships between them. In the context of word similarity models, they can be regarded as a means to compare two word spaces obtained from two different experiments.

In the past correlation analysis was generally used to measure correlation between variables. However it is dependent on the coordinate system in which the objects are represented. Even if there is a strong linear relationship between two sets of multidimensional variables, depending on the coordinate system used, it may not be visible as a correlation.

In order to remedy this problem, CCA transforms both the spaces so as to find pairs of subspaces which are maximally correlated with one another. In other words, CCA finds pairs of linear transformations of the two sets that are maximally correlated to each other.

CCA can be extended in an equivalent kernel form. Kernel canonical correlation analysis [21] extends CCA to find non linear relationships between two sets of variables. In the next section on KCCA, I have followed the discussion on KCCA by John Shawe-Taylor [21].

3.6 Mathematical Derivation of KCCA

Canonical Correlation Analysis can be defined as the problem of finding two sets of basis vectors, one for vector space **X** and the other for vector space **Y**, such that the correlations between the *projections* of the variables from these spaces onto these basis vectors are mutually maximized.

Let $(\mathbf{x}_i, \mathbf{y}_i)$ denote a sample of paired objects from the spaces X and Y respectively, where $\mathbf{x}_i \in X$ and $\mathbf{y}_i \in Y$. For instance, when CCA is used to compare two vector space representations of the same set of words w_i for $i = 1..m$, $\mathbf{x}_i$ and $\mathbf{y}_i$ would denote the representations of w_i in these two spaces.

Consider the case where only one pair of basis vectors $\mathbf{w_x}, \mathbf{w_y}$ ($w_x \in X$ and $w_y \in Y$) are sought, namely the ones corresponding to the maximal correlation.

By projecting $\mathbf{x}'_i$ and $\mathbf{y}'_i$ into w_x and w_y resepectively, we get the linear combination of the two variables as follows

$$x = \mathbf{x}'\mathbf{w_x} \text{ and } y = \mathbf{y}'\mathbf{w_y}$$

The function to be maximized is:

$$\rho = max_{\mathbf{w}_x, \mathbf{w}_y} correlation(x, y) \tag{3.44}$$

If we use $E[f(x, y)]$ to denote the empirical expectation of a function $f(x, y)$, we have:

$$E[f(x, y)] = \frac{1}{m} \sum_{i=1}^{m} f(x_i, y_i)$$

we can rewrite the correlation expression in terms of the empirical expectation as :

$$\rho = max_{\mathbf{w}_x, \mathbf{w}_y} \frac{E[xy]}{\sqrt{E[x^2]E[y^2]}} = max_{\mathbf{w}_x, \mathbf{w}_x} \frac{E[\mathbf{w}'_\mathbf{x}\mathbf{x}\mathbf{y}'\mathbf{w}_\mathbf{y}]}{\sqrt{E[\mathbf{w}'_\mathbf{x}\mathbf{x}\mathbf{x}'\mathbf{w}_\mathbf{x}]E[\mathbf{w}'_\mathbf{y}\mathbf{y}\mathbf{y}'\mathbf{w}_\mathbf{y}]}}$$

Now, the covariance matrix of $(\mathbf{x}, \mathbf{y})$ can be expressed in terms of the expectation as follows:

$$C(\mathbf{x}, \mathbf{y}) = E\left[\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}'\right]$$

This is equal to :

$$\begin{pmatrix} C_{\mathbf{xx}} & C_{\mathbf{xy}} \\ C_{\mathbf{yx}} & C_{\mathbf{yy}} \end{pmatrix} = C \tag{3.45}$$

The total covariance matrix C is a block matrix where the within set covariance matrices are $C_{\mathbf{xx}}$ and $C_{\mathbf{yy}}$ and the between sets covariance matrices are $C_{\mathbf{xy}} = C_{\mathbf{yx}}$

Hence we can write the function ρ as

$$\rho = max_{\mathbf{w}_x,\mathbf{w}_y} \frac{\mathbf{w}_x C_{\mathbf{xy}} \mathbf{w}_y}{\sqrt{\mathbf{w_x'} \mathbf{C_{xx}} \mathbf{w_x} \mathbf{w_y'} \mathbf{C_{yy}} \mathbf{w_y}}} \tag{3.46}$$

Now it can be observed that rescaling w_x or w_y either together or independently does not affect the above equation, so that for example replacing $\mathbf{w}_x$ by $\alpha \mathbf{w}_x$ gives the quotient

$$\frac{\alpha \mathbf{w}_x C_{\mathbf{xy}} \mathbf{w}_y}{\sqrt{\alpha^2 \mathbf{w_x'} \mathbf{C_{xx}} \mathbf{w_x} \mathbf{w_y'} \mathbf{C_{yy}} \mathbf{w_y}}} = \frac{\mathbf{w}_x C_{\mathbf{xy}} \mathbf{w}_y}{\sqrt{\mathbf{w_x'} \mathbf{C_{xx}} \mathbf{w_x} \mathbf{w_y'} \mathbf{C_{yy}} \mathbf{w_y}}} \tag{3.47}$$

Since the choice of re-scaling is therefore arbitrary, it is enough to try to optimize the following expression

$$\frac{\mathbf{w}_x C_{\mathbf{xy}} \mathbf{w}_y}{\sqrt{\mathbf{w_x'} \mathbf{C_{xx}} \mathbf{w_x} \mathbf{w_y'} \mathbf{C_{yy}} \mathbf{w_y}}} \tag{3.48}$$

subject to:

$$\mathbf{w}_x' C_{\mathbf{xx}} \mathbf{w}_x = 1 \tag{3.49}$$

$$\mathbf{w}_y' C_{\mathbf{yy}} \mathbf{w}_y = 1 \tag{3.50}$$

This is the standard CCA model. However at this point, the *dual form* of the above expressions can be derived using the definition of $C_{\mathbf{xx}}$ and $C_{\mathbf{yy}}$ and the data matrices (of vectors) X and Y. The idea of expressing in the dual form is to incorporate kernels into the standard CCA model.

We have:

$$C_{\mathbf{xx}} = X'X$$

and

$$C_{\mathbf{xy}} = X'Y$$

The directions $\mathbf{w}_x$ and $\mathbf{w}_y$ can be rewritten as a linear combination of data vectors into the direction of α and β.

$$\mathbf{w}_x = X'\alpha$$

$$\mathbf{w}_y = Y'\beta$$

where α and β are vectors.

Substituting into the CCA expression, we obtain

$$\rho = max_{\alpha,\beta} \frac{\alpha' XX'YY'\beta}{\sqrt{\alpha' XX'XX'\alpha \cdot \beta' YY'YY'\beta}} \tag{3.51}$$

If $K_x = XX'$ and $K_y = YY'$ be the kernel matrices corresponding to the two representations, we substitute into the above equation to obtain

$$\rho = max_{\alpha,\beta} \frac{\alpha' K_x K_y \beta}{\sqrt{\alpha' K_x^2 \alpha \beta' K_y^2 \beta}} \tag{3.52}$$

At this point the norms of the associated weights are penalized. The reason for this penalization is that if K_x and K_y are full rank, then for any direction w_x, there is a direction w_y for which the correlation is perfect. An detailed discussion can be found in John Shawe-Taylor's Kernel methods for pattern analysis [20]. The penalization results in :

$$\rho = max_{\alpha,\beta} \frac{\alpha' K_x K_y \beta}{\sqrt{(\alpha' K_x^2 \alpha + k\|w_x\|^2) \cdot (\beta' K_y^2 \beta + k\|w_y\|^2)}}$$

$$\rho = max_{\alpha,\beta} \frac{\alpha' K_x K_y \beta}{\sqrt{(\alpha' K_x^2 \alpha + k\alpha' K_x \alpha) \cdot (\beta' K_y^2 \beta + k\beta' K_y \beta)}} \tag{3.53}$$

Thus the KCCA optimization problem is the maximization of the numerator of equation above equation subject to the following constraints:

$$\alpha' K_x^2 \alpha + k\alpha' K_x \alpha = 1$$

$$\beta' K_y^2 \beta + k\beta' K_y \beta = 1$$

The corresponding Lagrangian is

$$L(\lambda_\alpha, \lambda_\beta, \alpha, \beta) = \alpha' K_x K_y \beta - \frac{\lambda_\alpha}{2}(\alpha' K_x^2 \alpha + k\alpha' K_x \alpha - 1) - \frac{\lambda_\beta}{2}(\beta' K_y^2 \beta + k\beta' K_y \beta - 1)$$

Taking derivatives with respect to α and β, we obtain

$$\frac{\partial f}{\partial \alpha} = K_x K_y \beta - \lambda_\alpha (K_x^2 \alpha + k\alpha' K_x \alpha) = \mathbf{0} \tag{3.54}$$

$$\frac{\partial f}{\partial \beta} = K_y K_x \alpha - \lambda_\beta (K_y^2 \beta + k\beta' K_y \beta) = \mathbf{0} \tag{3.55}$$

Subtracting β' times the second equation from α' times the first, we have

$$\begin{aligned}0 &= \alpha' K_x K_y \beta - \alpha' \lambda_\alpha (K_x^2 \alpha + k K_x \alpha) - \beta' K_y K_x \alpha + \beta' \lambda_\beta (K_y^2 \beta + k K_y \beta) \\ &= \lambda_\beta \beta' (K_y^2 \beta + k K_y \beta) - \lambda_\alpha \alpha' (K_x^2 \alpha + k K_x \alpha)\end{aligned}$$

which together with the constraints imply that $\lambda_\alpha - \lambda_\beta = 0$. Let $\lambda = \lambda_\alpha = \lambda_\beta$.

Considering the case where the kernel matrices are invertible, we have

$$\begin{aligned}\beta &= \frac{(K_y + kI)^{-1} K_y^{-1} K_y K_x \alpha}{\lambda} \\ &= \frac{(K_y + kI)^{-1} K_x \alpha}{\lambda}\end{aligned}$$

and so substituting in equation 3.54 gives

$$K_x K_y (K_y + kI)^{-1} K_x \alpha - \lambda^2 K_x (K_x + kI) \alpha = 0$$

Hence

$$K_y (K_y + Ik)^{-1} K_x \alpha - \lambda^2 (K_x + kI) \alpha = 0$$

or

$$(K_x + kI)^{-1} K_y (K_y + kI)^{-1} K_x \alpha = \lambda^2 \alpha$$

This is a generalized eigenvalue problem of the form $Ax = \lambda x$. The above equation can be solved to obtain the required basis vectors.

I have used Gaussian kernels with KCCA for comparing our word space and that of Schütze [16]. This is described in greater detail in the next chapter.

Chapter 4

Experiments on similarity modelling with slot phrases

4.1 Introduction

The objective of this chapter is to introduce and develop a computational model of word similarity using slot phrases. Chapter 2 introduced slot phrases and described the steps involved in constructing a vectorial representation of words: the *slot-word matrix.* The slot word matrix is used to construct a low dimensional representation: *word space.* This chapter describes the analysis of this word space constructed using general and specific slot phrases. The latter is then compared to Schütze's low dimensional representation of words.

The slot-word matrix which is the high dimensional sparse matrix, represents words in terms of their frequency of occurrence in various slots. Using this matrix, two words are similar if they tend to share the same set of slots and two slots are similar if they tend to share the same set of words. However, two slots may be similar because they may share a *similar* set of words. Consequently, two words may be similar because they tend to share a *similar* set of slots.

Therefore, if word $w1$ and $w2$ share some slots, and word $w1$ and $w3$ share many similar slots(but not many same slots), it may be the case that $w1$ is more similar to $w3$ than to $w2$. This kind of indirect similarity brought about by inductive inference is called *latent similarity.* Clearly, the slot-word matrix does not capture latent similarity between words as the latent relationship between $w1$ and $w3$ is not captured in this matrix.

This idea was illustrated in chapter 3, where it was shown that using

SVD, it is possible to capture the latent similarity between words. Using a synthetic example, it was illustrated how a space which captures latent similarity often reflects the true relationship between words.

SVD is therefore performed on the slot-word matrix to obtain a lower dimensional matrix as explained in the previous chapter. The columns of this matrix represent the words as low dimensional vectors. This low dimensional space will be reffered to as the *word space*. SVD is used with the expectation that the resulting word space would capture the true similarity relationship between words.

This word space is the desired similarity model using slot phrases as a context. This distribution of words in this space can be analyzed, as words are represented as vectors, making it possible to compute similarity and distance.

The objective of the experiments in this chapter is to first investigate the word space and then compare this word space with other low dimensional word representations which were induced by different definitions of context.

While the dimension of the word space is relatively low (25), it is still too high for people to be able to visualize directly and understand. We therefore carried out several exploratory experiments to understand this word space and examine whether this space has captured word similarity. These are carried out first for the word space obtained using general slot phrases and then for the word space obtained using specific slot phrases.

This word space obtained using specific slot phrases is then compared with a representation obtained by Schütze's [16] method. KCCA and nearest neighbors of selected words are used for the comparison of these two spaces. Schütze's method is used for comparison because this seems to be the method that most closely resembles our methods.

The next section describes how a word space is obtained from the slot-word matrix using singular value decomposition (SVD).

4.2 From Slot-Word matrix to Word Space using SVD

4.2.1 Introduction

This section deals with the procedure to obtain a lower dimensional word space using SVD on the slot-word matrix. The columns of slot word matrix $A(m \times n)$ represent words in terms of their frequency of occurrence in slots. The entry $a_{i,j}$ of this matrix contains the frequency of the word j in slot i.

Singular value decomposition is applied to the slot-word matrix, resulting in three matrices U, S and V.

The columns of U are the orthogonal set of eigenvectors of the matrix $A'A$– the word-similarity matrix obtained by the dot product of the word vectors with each other. These eigen vectors are therefore called "eigen words".

A small number of k eigen words are chosen and the original word vectors (the columns of the slot-word matrix) are projected into each of the k eigen words to obtain k values for each word.

These k values for each word vector are the coordinates of each word in the k dimensional space. This k dimensional space is the lower dimensional *word space*– the desired word similarity model. The following section describes these steps in greater detail.

4.2.2 Practical steps to obtain the word space using SVD

- **Aim:** To perform SVD on the slot-word matrix to obtain the word space.

- **Method:**

  1. The slot-word matrix A is first whitened in two steps:
     (a) First, by multiplying the columns by $log\frac{n}{n_j}$, n being the total number of slot phrases and n_j is the number of slot phrases containing the word j. This multiplication gives greater weight to words that are more specific to some slot phrases and lesser weight to those words that are more widely distributed in the slot phrases. This method is similar to that which is used in the document retrieval literature for TF/IDF scaling.
     (b) Second, normalizing the rows by dividing each row $\mathbf{A_i}$ by $\frac{1}{(\sum_{j=1}^{n} a_{i,j}^2)^{\frac{1}{2}}}$. root of the sum of squares of each attribute of that row.
  2. Singular Value Decomposition is performed. It results in U,S and V.
  3. The words (columns of the Data matrix) are then projected into the first k columns of U. The projection is calculated as $S \cdot V'$ for the words, as explained in the third chapter.

4. The projections obtained in the previous step are divided by the singular values. This results in V'. This division is performed in order to equalize singular values, so that the distances are not excessively skewed towards some directions.
5. The columns V' are the desired word vectors.

The above procedure results in the desired low dimensional word space which needs to be examined and analyzed. Prior to analyzing this matrix, it would be useful to find the number of dimensions k for which the nearest neighbors to a given set of words are most similar.

Optimal dimension of the word space

The choice of an appropriate number k of dimensions partly a matter of judgement. The dimensions with large singular values are likely to be meaningful, and dimensions with small singular values are likely to be increasingly affected by noise. We conducted a small investigation to choose an appropriate number of dimensions to keep. To do this we have examined the nearest neighbors of the words for different values of k, for different words to see at what value of k, these nearest neighbors are intuitively most similar to the words.

- **Aim:** To find out an appropriate value for k–the number of dimensions of the word space, for which the nearest neighbors of 5 words seem most intuitively similar to it.
- **Method**:
  1. For different values of k (the number of dimensions of the word space) a list of the 50 nearest neighbors is first listed for each of the 5 words in the first column of Table 4.1.
  2. For each of these 5 words, the number of similar words are counted for each value of k and listed as rows of the table. For instance the first row lists the number of similar words to Brazil i.e. country names, for $k = 5, 10, 15..$ etc.
  3. The sum of the values for each column (each k) is calculated and added as the last row of this table. The first entry of the last row (Total) denotes the total number of similar words to the 5 words for $k = 5$. The last row can therefore be considered as a representative sum for the number of similar words for each dimension k.

$k =$	5	10	15	20	25	30	40	60	100
Brazil	11	20	21	21	24	33	34	28	25
Grandfather	8	8	20	20	30	18	23	14	11
Julia	16	31	39	38	45	37	30	29	25
Laugh	1	2	3	21	20	15	18	15	24
Surprised	5	9	15	16	22	20	14	17	12
Total	41	70	98	116	**141**	123	119	103	97

Table 4.1: Number of similar words to 5 words for each value of k

4. The last row is examined to see at which dimension, there are most number of intuitively similar words.

- **Conclusion**: From the last row of table 4.1, it can be seen that the largest number of intuitively similar words for the 5 words occurs when $k = 25$. The total number of similar words is 141 as given in the table. Therefore a reasonable choice of a dimension of the word space is 25. Therefore, in the rest of the study, I have used SVD to obtain a word space of 25 dimensions.

4.3 Finding Semantic and Syntactic Information in the Lower Dimensional Word Space: A summary

4.3.1 Introduction

The next logical step is to examine this space for patterns and understand the distribution of words in this 25-dimensional space. Therefore the first set of experiments are exploratory. Having explored this space, the next set of experiments aim at comparing this word space to Schütze's space. The following is a roadmap of the experiments in this chapter.

- **Exploratory Analysis of the word space** The first three experiments explore the word space obtained using general slot phrases and the fourth one deals with the word space obtained using specific slot phrases. These are summarized below.

  1. Nearest Neighbors The first experiment examines the nearest neighbors of some selected high frequency words. The aim of this experiment is to see whether these nearest neighbors are similar

to a given word and if so in what ways they are similar. This experiment is described in section 4.4.1.

2. Clustering Nearest neighbors provides information about a very small fraction of the whole space. In order to understand the distribution of words and examine whether clusters of words exist, k-means clustering is performed on this space. The second experiment is described in section 4.4.2.
3. Support Vector Machines k-means clustering is seen to yield separate clusters of the same category of words (such as adjectives). It is not clear whether these separate clusters of the same categories are together or apart. In other words, it is not clear whether all the words of the same category such as adjectives can be captured in a region. The third experiment investigates whether it is possible to capture words in a region using a SVM with an RBF Kernel.
4. The word space using specific slot phrases This experiment generates a word space using *specific slot phrases* and compares this space with the word space obtained using general slot phrases. Section 4.5 describes this experiment.

- **Comparisons with other word spaces** This final experiment compares the results of the specific word space with those of Schütze in section 4.6.

The rest of the chapter describes each of these experiments in greater detail.

4.4 Exploratory analysis of the word space

4.4.1 Nearest Neighbors for Selected Words in the general slots word space

The next two experiments aim to explore the word space using nearest neighbors of selected words. The word space is obtained using general slot phrases and has 25 as the dimensionality. The first experiment examines how much syntactic similarity is captured in this space. The next experiment looks at whether this space has captured semantic similarity.

Syntactic similarity in the word space

- **Aim**:The aim of this experiment is to find out wither the nearest neighbors of word in the word space are syntactically similar or "Does closeness imply syntactic similarity?".

- **Method**:

  1. 20 words are selected at random from a list of 10000 words. From these, 5 words that occurred with several different syntactic forms are chosen. These 5 words happened to be verbs whose roots are amuse, address, admire, adore and describe.
  2. For each of root words, their different syntactic forms are tabulated in the first column of table 4.2. The nearest neighbors for these words are listed in the rows of the table 4.2. For instance, the first four row lists the nearest neighbors for *amuse, amused, amusement* and *amusing* which are different grammatical forms of the word *amuse.*
  3. The nearest neighbors are examined to see whether they are of the same grammatical category– same tense, same part of speech and the same number– as the words for which the nearest neighbors are considered. For instance, it can be observed that the nearest neighbors of the word *amuse* are verbs in the present tense. Therefore a score of 100% is given to POS and tense(in column 4 and 5) for this word.
  4. The percentage of nearest neighbors, which have the same part of speech(POS), tense and number[1] is calculated for each word. This is recorded in the 3rd, 4th and the 5th column of the table.

- **Conclusions**:

  1. Same part of speech: On an average, 98.5% of the neighbors can be interpretted as the same part of speech. The words have been considered in isolation rather than in context. Nevertheless, the agreement is striking. This implies that the word space has captured "part of speech similarity".
  2. Same Tense: 96% of the Verbs are of the same tense as the words considered. This implies that the word space has captured "Tense similarity".

[1]tense and number are evaluated where applicable

Words	Nearest Neighbours for the Word	POS	Tense	No.
amuse	rouse, injure, bid, inspire, impose pursue, urge, adore, invite, remind, recognize haunt, deceive, break, puzzle, execute, curse	**100**	**100**	
amused	impressed, concealed, assailed, emptied thrilled, questioning, reassured, lowered supported, touched, conveyed, fortified, choked	100	95	
amusement	governments, libraries, tickets culture, trials, logs, operations, centuries naval, worlds, organisms, glaring, remedies, kings	100		
amusing	excellent, entertaining, dangerous interesting, serious, important, funny, wonderful impressive, seriously, satisfactory, valuable	95		
address	temper, name, position, haunts, employer apprehensions, career, ears, favorite, lips, identity interests, squaw, grandson, owners, commission	100		90
addressed	laid, sold, indicated, opened buried, served, judged, delivered, placed, struck kept, carried, despised, hung, burnt, arranged	100	100	
admire	seize, salute, shake, weed, bough, tie commence, pursue, divide, dislike, carried, practise ben, qui, puzzle, recognize, abuse, expose	90	90	
admired	admired, prevented, smashed, released, disturbed killed, blinded, taught, attacked, solved, ignored secured, fed, avoided, adopted, paid, inserted	100	100	
admirable	rare, obvious, ridiculous, wonderful high, beautiful, splendid, partial, clear, doubtful delicate,pleasant,suggestive,entertaining,romantic	100		
admirably	briefly, quietly, rarely, slowly, lightly softly,earnestly,brightly,sharply,sincerely,hopelessly faintly,properly,dost,graciously,calmly,reluctantly	95		
admiration	rage, charity, dignity, anxiety malice, bottles, dew, excitement, indignation,pearls contempt, courtesy, wisdom, approbation, aversion	100		90
adore	betray,bid,console,serve,greet,recommend devor,remind,gratify,distinguish,execute,burn break,prepare,punish,accept,lose,fill	100	100	
adored	whirled,smote,prompted,possessed threw,kicked,disliked,embraced,counted,receives slew,shook,gave,chased,describes,nursed,hired	100	90	
adorations	woes,manners,thoughts,tastes,impatience cruelty,courage,experiments,instructions,actions projects,bravery,clothes,conclusions,labors,threats	100		95

described	developed,discovered,robbed,invented mended,produced,gained,attained,cleared,won paid,committed,sustained,forgotten,mentioned	100	95	
describing	avoiding,crossing,conquering,using taking,clapping,saving,searching,mentioning finding,similarly,whereby,expecting,acknowledged	95	90	
description	sense,explanation,recent,conspiracy object,incident,color,person,impossibility,class quarter,idea,resolution,objection,gentleness,gift	85	90	
descriptions	exclamations,communications differences,matters,speculations,privileges	100		100
Average%	POS, tense and number:	97	96	91

Table 4.2: The nearest neighbors for 5 random words and their syntactic forms.

3. Same Number- Singular vs Plural: 91% of the nearest neighbors of the nouns are of the same number. This implies that the word space has captured similarity based on number.
4. Semantic Similarity: By looking at the nearest neighbors of the words in the table, it can be concluded that except for the word *admirable* most of the nearest neighbors are **not** semantically close. This implies that the word space has not successfully captured semantic similarity for these 19 random words.

- **Discussion** The results of this experiment show that the word space obtained using general slot phrases has captured aspects of syntactic similarity well. The results are negative for semantic similarity. However, the 5 words chosen were random and the case may be that the word space has captured semantic similarity for certain typical words (explained below). This aspect is examined in the next experiment.

Semantic similarity in the word space

The aim of the following experiment is to investigate whether the word space obtained by using general slot phrases has captured semantic similarity for at least a small number of words typical of natural semantic categories(I have reffered to these words as "typical words").

I have therefore chosen typical words from the list of 10000 words (This is the list of words for which the word space has been constructed). These words are typical in the sense that the words that are semantically close to these words appear in this list. For instance, this list contains at least 500 country names and people's names. Therefore, if the word space has captured some semantic similarity, it should at least group these natural categories such as names together. This can be known by looking at the nearest neighbors for words from these natural categories.

- **Aim**: The aim of this experiment is to understand to what extent the word space has captured semantic information, by an examination of the nearest neighbors for some typical words.
- **Method**:
  1. Examples of five very common categories of words are selected. The five categories are surnames of people, names of countries, the months, ordinal numbers and the days of a week. One example for each category is chosen. The examples were *Alleyene, Africa, April, seventh* and *Monday.* 15 nearest neighbors of these words are found and listed in table 4.3
  2. Next, five common categories of words are chosen, on the basis that the family of words relating to these categories are all well represented in the in the list of 10,000 words. For instance the 10,000 word list contains most of the body parts. The categories chosen are body parts, professionals, type of animal, bodies of water and a insulting word. The examples for these categories were *eyes, captain, bird, canal* and *fool.* The nearest neighbors for these words are found and listed in table 4.4
  3. The nearest neighbors are examined to see whether they are semantically similar to 10 chosen words.
- **Conclusions**:
  1. Looking at the nearest neighbors of very common words such as names, months and days, it can be seen that some neighbors are semantically similar to the these words. It may be concluded that the word space has captured semantic similarity for such very typical and common words.
  2. It can be observed that the while some of the neighbors of the typical category of words in table 4.4 are semantically similar, most of them are not similar semantically.

Words	20 Nearest Neighbours for Very Common Words
alleyne	dorothy, 'good, holmes, katy, sancho, linda, spilett, betsy, toots, boffin, fledgeby, cyril, pancks, ozma, wyatt, anthea, donald, ojo, bradley,
africa	spain, england, burgundy, chicago, sweden, holland, france, portugal, paris, london, richmond, australia paradise, maine, persia, jerusalem, christendom, greece
april	july, february, october, june, december,january august, september,november,kent,france,vienna,egypt, england, peru, syria, texas, london, commerce,
seventh	sixth, eleventh, fifth, twelth,youngest,biggest noblest, smallest, muffats, austrian, outer, sixteenth
monday	tuesday,march,thursday,saturday, scooter car,friday,michael,evening,sunday

Table 4.3: Nearest neighbors for very common words

Words	20 Nearest Neighbours for some typical categories
eyes	`fingers`, papers, `hands`, toilet, sister's, `jaws`,husband `legs`, chum, kinsmen, displeasure, `lips`,jaundice, computer, `neck`, `cheeks`, shoulders, wife's, `nerves`
captain	colonel, engineer, manager,virginian abbe, marquise, corporal, count, doctor, bishop, major, curate, baroness, typewriter, duke,green,scarecrow
birds	women, insects, courts,animals,streams,purity cats, cattle, flags, passengers, schools,cows,kings logs, plants, spirits, races, fowls, bacon
canal	admiralty,crater,galley,river,seals,lake chimney,gentiles,hive,gospel,crowd,grasses,scriptures abyss,ocean,mountains,saloon,hebrews,underground
fool	coward,flash,cock,woman's,liar,beggar madman,monkey,mason,wolf,policeman,bomb,clown mouse,child's,screen,panther,dog,tiger

Table 4.4: Nearest neighbors for typical categories

- **Discussion:** This experiment and the previous one suggests that the word space has captured very limited semantic similarity. It has been able to capture semantic similarity only for well defined categories of words such as names. This is not surprising as these names, months and days have very standard usage patterns.

  On the other hand, most of the nearest neighbors of even very natural categories such as body parts are not semantically very similar. This is despite the fact that all the closest words in these categories are present in the list. For instance the 10000 word list contains all the body parts. But the word space has **not** been able to group all these semantically close words together.

  The reason why general slot phrases are more inclined to capture syntactic rather than semantic similarity can be intuitively explained. Consider the slot phrase "the X". This slot phrase suggests that X is mostly a noun or adjective. In other words this slot phrase conveys syntactic similarity rather than semantic similarity.

  This analysis suggests that semantic similarity may be better captured using *specific slot phrases.* This idea is developed in section 4.5 on Specific slot phrases. The next two sections aim at further exploring the slot phrase obtained using general slot phrases.

4.4.2 Clustering in the Word Space

The nearest neighbors approach has given an idea about the distribution of words around specific words in the word space. More precisely it has conveyed that in the word space, nearness implies syntactic similarity and less often, semantic similarity.

However, the nearest neighbors approach has considered a small sample and therefore does not give a bigger picture about how the words are distributed. For instance, the nearest neighbors approach does not tell us whether there are some dense clusters of similar words in the word space and if so, what they are.

The following experiment uses the k-means clustering algorithm to identify k dense centers in the word space and examines the k partitions around these centers.

Finding Clusters using the k-means Algorithm

- **Aim:** To cluster the Words using the k-means Algorithm to examine whether there are distinct groups of similar words

- **Method:** k-means Clustering with k=100 is performed on the word vectors in the word space. The choice of $k = 100$ is heuristic and this value has been chosen on the basis that when $k = 100$, the number of words per cluster is around 100(total number of words being 10,000). Experimental results with various values of k show that when k is less than 100, semantically similar words are less likely to appear in the same cluster.

  The k-means algorithm described in the previous chapter in section 3.3.1, is used for clustering the word space. The means are intialized with 100 randomly chosen word vectors. The k-means algorithm then lists the 100 clusters of 10000 words in a file. Table 10 and 11 contain some typical clusters from the 100 clusters sets from this file.

- **Conclusions**:

  - Semantic Clusters: The following are the semantic clusters found in the word space:
    1. Clean clusters of the names of people and places are found in the word space. Table 4.5 lists one such cluster of people's and place's names[2].
    2. Another clean cluster is that of words that refer to people's profession, nationality etc. The fourth row of Table 4.5 lists a sample of such a cluster.
    3. The word space contains many mixed clusters. For instance the body parts and family relatives are clustered together in the word space as illustrated in the table.
  - Syntactic Clusters: An extremely interesting outcome of the k-means clustering is the presence of clean syntactic clusters with hardly any noise in them.The following are some typical syntactic clusters found in the word space:
    1. Words of the same parts of speech are found in clean clusters with very few exceptions. Table 4.6 lists one such clean cluster of adjectives.

[2]It can be observed that name of months has appeared along with the names of places.

Type	*Clusters*
Contemporary Names	adrian, alfred, ali, allan, amanda, andrea, anne, aramis, arthur, athos, baartock, bella, bernenstein, bertram, bickley, bland, caderousse, capt, carroll, catherine, charley, clennam, cleopatra, danglars, darwin's, desdemona, dinadan, dolly, dr, edith, eugene, fanny, ferdinand, fouquet, heidi, henry, hoopdriver, isabelle, ivan, jane, jasmin, jason, jerome, jim, joe, jones, jukes, k, kay, kerry, kitty, koku, laguitte, lassiter, levin, ling, lucien, m, madame, margaret, martin, max, milady, mme, modeste, monte, monty, morrel, mr, mrs, muller, myles, ned, nelson, newmark, oscar, paul, peter, piles,
Old Names	achilles, aeneas, alexander, analogy, burke, c, caesar, cambridge, clarence, comprehension, constantinople, coriolanus, cromwell, daniel, degrees, drowning, e, everybody, everyone, exeter, feudal, florence, foreigners, geology, glaucus, harold, hastings, moses, murray, oliver, pasture, pharaoh, plato, rachel, routine, s, satan, saul, sinners, sparks, st, suffolk, suicide, sylvie, thence, timon, usage, venus, wright, xenophon,
Body Parts and Relatives	address, adventurous, adversary, antagonist, attachment, aunt, beard, belt, bonnet, bosom, brother, career, cheeks, chum, collar, commandments, companion, complaints, confederates, counsels, countenance, cousin, cousins, daughter's, descent, dinners, displeasure, dress, ears, face, favor, fingers, fits, foes, gait, hair, heels, hind, husband, jaws, kinsmen, legs, lips, mother's, mouth, nails, name, neck, neighbor, niece, nostrils, owners, pallid, plight, pockets, pulse, pupils, reputation, squaw, stockings, temper, temples, toes, toilet, tones, tongue, tracks, travels, voice, wife's,
Occupation	abbe, abbot, actress, apeman, barber, baron, baroness, biggest, bishop, boss, breeze, bride, bridegroom, bug, butler, canon, captain, cardinal, chevalier, chicken, chief, clerk, coachman, colonel, colonial, conductor, constable, cook, corporal, count, cowboy, curate, hermit, housekeeper, inspector, jew, journalist,knight local, magician, maiden, major, manager, marquise, miller, monk, pastor, phonograph, pilot, porter, president, prince, princess, professor, queen, raven, reporter, sailor, scarecrow, scout, seaman, secretary, senior, shepherd, sheriff, singer, skipper, steamer, steward, stranger,trader, traveller, widow,landlady, landlord, lieutenant,
Places Months and Other Nouns	abundance, africa, albany, alms, america, anticipation, april, asia, august, australia, austria, banishment, battle, bliss, brass, brazil, burgundy, california, charles, chicago, china, christendom, christianity, civilization, civilization, commerce, contentment, december, england, europe, feb, february,france,georgia normandy, november,october,pennsylvania, persia, philadelphia, poland portugal,rochester, russia, scotland, september, spain, sweden

Table 4.5: Some typical semantic clusters

Type	*Clusters*
Adjectives	big, black, bright, cheerful, cold, constant, cowardly, critical, cunning, deadly, deep, dense, dignified, dim, dull, endless, fearful, fierce, fragrant, gay, heavy, heroic, humorous, irregular, jolly, joyous, legitimate, lofty, mild, noble, noisy, peculiar, penetrating, precise, primary, profane, rough, sharp, slow, stately, strong, tender, tense, vague, valiant, white, wide, wild,
Mixed clusters of adverbs and Adjectives	accidentally, affectionately, angrily, awfully, barbarous, bitterly, blond, brightly, carefully, cautiously, characteristic, cheerfully, confidently, convincing, cordially, cruelly, curious, decidedly, deliberate, deliberately, destitute, differently, difficult, distinctly, earnestly, electronically, extraordinary, fairly, flattering, fortunately, foul, freely, gently, gladly, graciously, gratefully, heartily, highly, horrible, humbly, important, imposing, incredible, instructive, interesting, largely, leisurely, likely, literally, loudly, nasty, nice, nicely, nobly, noiselessly, offensive, orderly, politely, possibly, potent, probably, punitive, quietly, rapidly, remarkably, resolutely, respectfully, sadly, seldom, seriously, severely, sincerely, singular, slowly, smart, softly, solemnly, strange, strangely, strongly, tenderly, txt, unexpectedly, unusual, vividly, widely, willingly, wisely,
Present Tense	abuse, admire, afford, bad, ben, bin, bough, capture, carrie, commence, curve, dam, devise, dislike, divide, doe, don, double, drive, enter, expose, fer, haunt, hid, impose, injure, inspire, kai, le, lea, loose, manage, nurse, pace, practise, praise, produce, pronounce, pursue, puzzle, qui, raise, recognize, remove, resume, revive, rouse, salute, se, seize, shake, tie, urge, waste, wave
Present Continuous	'at, 'in, 'n', 'of, 'to, 'where, accepting, adjoining, an', announcing, arched, assuming, avoiding, bearing, beating, beholding, believing, catching, choosing, clapping, clasping, commanding, concluding, conquering, containing, counting, covering, cracking, crossing, declaring, describing, distorted, dividing, dropping, ere, expecting, finding, giving, hearing, helps, hiding, il, improving, indicating, kicking, laying, leaving, letting, lifting, lowering, measuring, mentioning, noticing, notwithstanding, opening, ordering, perceiving, picking, placing, possessing, raising, remembering, representing, requiring, resembling, rubbing, saving, scanned, searching, seizing, showing, similarly, sounding, stating, suggests, taking, tapping, tearing, throwing, thrusting, touching, using,
Past Tense	announced, answer'd, answered, answers, asked, assented, canto, chapter, commented, con, continued, cried, cries, daresay, declared, demanded, ejaculated, exclaimed, explained, faltered, gasped, groaned, growled, ha', hasn't, headlong, inquired, knit, knowed, leaps, murmured, mused, muttered, nodded, objected, paced, paused, pleaded, prithee, protested, qualified, queried, quivered, quoth, rejoined, remarked, repeated, replied, responded, resumed, retorted, says, screamed, shrieked, shrugged, sighed, snapped, sobbed, stammered, stooped, ventured, whispered

Table 4.6: Some typical syntactic clusters

2. Verbs of the same tense are clustered together. The table lists present, past and present continuous as samples of these verb-tense clusters.
3. The nearest neighbors of the word on the 4th row of the table are all largely of the same number. This suggests that the space has also captured similarity based on number.

- **Discussion:** The results of the above experiment support and explain the results of the nearest neighbors experiment according to which the word space has captured syntactic similarity extremely well. These clear clusters suggests that the word space consists of many clean clusters and a few mixed *syntactic clusters*.

  However this experiment does not convey whether different clusters of the same syntactic category are found together in the space. Also, the experiment uses hyperellipsoids to separate the clusters. The separate categories such as adjectives and adverbs present together in mixed clusters may not be separable using simple hyperellipsoids. Therefore the question whether these can be separable, atleast by a more complex region arises. These two aspects are examined in the next experiment.

4.4.3 Using SVM to find similar words

k-means clustering of the word space has shown that there are several clear clusters and some mixed clusters of words belonging to the same syntactic category (such as several clusters of adverbs). However, it is not clear whether several clusters of the same category are found together in the space. Moreover, several *mixed clusters* were found in the previous experiment suggesting that it was not possible to separate these categories using simple hyperellipsoids used by the k-means algorithm. This analysis raises the following question: "Is it possible to capture words of the same syntactic category in a region?"

The following experiment attempts to answer this question using support vector machines (SVM). SVM is a well known classifier which constructs a hyperplane separating the positive and negative examples in a training set. SVM is known to generalize well on the test set. Moreover, kernels can be incorporated into the SVM by expressing the SVM classification problem in the dual form. kernels make it possible to separate data which are not linearly separable by a hyperplane. RBF kernels are special types of kernels known for their ability to form connected islands or pools around the positive data points.

Type	Perc.Accuracy for +ve Test Data	Perc. Accuracy for -ve Test Data
ly-words	90	95.6
ing-words	92	98
ed-words	90	95

Table 4.7: Classification results of the SVM

In the experiment described below, I have tried to examine whether adverbs, verbs in present tense and past tense can be captured within a region using SVM with a RBF kernel.

- **Aim:** To capture adverbs, verbs in present and past tense within a region using SVM with a RBF kernel. SVM
- **Method:**
  1. Three sets of positive and negative examples are first constructed. The positive examples of the first set consists of all the 'ly' ending adverbs. The corresponding negative examples are non adverbs. The positive examples for the second set are all the verbs ending with 'ed'. The negative examples for this set are the rest of the words in the 10000 word list. 'ed'. The third set of positive examples consists of verbs ending with 'ing'. The corresponding negative set of examples consists of the rest of the words in the 10000 word list.
  2. A training set is constructed for each of these three sets (ly-adverbs, ed-verbs and ing-verbs) by selecting 50 positive examples and 50 negative examples at random from the set of positive and negative examples constructed in the previous step.
  3. A test set is constructed for the three sets by selecting 800 positive and 2000 negative examples. The set of adverbs not ending with 'ly' are added to the test set. This is to see if the region trained on 'ly' ending adverbs also captures the rest of the adverbs.
  4. The SVM is trained on the training set.
  5. The SVM is then tested on both positive and negative examples from the test set. The result is tabulated in Table 4.7.
- **Conclusions**: It can be seen from the results that the classification accuracy is quite high for all the three types of words considered. This

implies that the for these three categories, it is possible to capture the words of those categories in a region using a RBF kernel.

- **Discussion:** The above experiment has demonstrated that it is possible to capture words of a particular syntactic category in a region using an SVM with a RBF kernel. However, this analysis is not exhaustive and needs to be carried over to other syntactic categories found from the clusters obtained by the k-means clustering algorithm.

4.4.4 Experiments with specific slot phrases

It was observed in the previous experiments that the word space constructed using general slot phrases captures syntactic rather than semantic similarity.

The next two experiments investigate the effect of using more specific slot phrases. A method to generate specific slot phrases from the general slot-word table was described in chapter 2. Specific slot phrases were constructed by substituting repeated words in their respective slots to create more specific slot phrases. However, it was noted that while a whole spectrum – from general to very specific slot phrases– can be created using the method, I have at present myself to constructing only one level of specificity due to limitations in time for coding. Therefore the present code needs more extensions to realize different levels of specificity.

In the first experiment, I have constructed a word space by using specific slot phrases and compared with the word space obtained from general slot phrases. The two spaces are compared by comparing the nearest neighbors for 10 selected words, out of which 5 words are typical words and 5 words are random high frequency words.

The second experiment examines the nearest neighbors for very rare words using specific slot phrases.

Comparison of general and specific slot phrases

- **Aim**: To examine the effect of making slot phrases more specific by comparing the nearest neighbors of 10 selected words, obtained using the general and specific slot phrases respectively.

- **Method:** The following experiment can be divided into three parts:1) Generating specific slot phrases 2) Constructing the corresponding word space and 3)Comparing this word space with the word space obtained using general slot phrases. The following passages describe each of these three parts.

1. Generation of specific slot phrases Specific slot phrases are generated from the entries in the slot word table(obtained using general slot phrases). From this table, slot phrases with more than 2 slots are considered. A typical entry in the slot word table may look like this:

   I went to the X1 to X2:`stadium:jog+club:jog+park:jog+stadium:exercise....`

   (a) Words that occur in the second position of atleast 3 times are identified. In the above example "`jog`" appears three times and it is found in the second position. For the moment, I have limited myself to considering the frequent word in the second position.
   (b) A *specific slot phrase* is constructed by substituting the frequent words(`jog` in this case) in its corresponding slot X2 in the slot phrase entry.In this example the general slot phrase is:

       I went to the X1 to **X2**

       and the corresponding specific slot phrase obtained by substituting X2 for **jog** is

       I went to the X1 to **jog**

   (c) The words that appear with this frequent word (`Jog`) such as `club, park` and `stadium` are then extracted and a new slot-word entry is formed for the specific slot phrase. This specific slot-word entry is added to the general slot phrase table. In this case, the specific slot-word entry would be:

       I went to the X1 to **jog**:- `club:park:stadium`.

       Note: when there are three or more slots in a slot phrase, there may be repeated words in the specific slot phrase generated in the previous step. Therefore, more specific slot phrases would have to be generated from these specific slot phrases using a similar procedure as described above. However, at this point, due to time constraints, I have limited myself to just one level of specificity.

   (d) Steps a,b,c are repeated for every entry of the *general-slot-word-table* and the new specific slot-words entries are together added to the general slot-word table. This results in a bigger slot-word table containing both general slot phrases and specific slot phrases.

2. <u>Word space from Specific Slot Phrases</u> The slot word table prepared using the procedure above is used to construct a word space in the following steps:

   (a) This extended slot-word table is used to create a slot-word matrix using the procedure described in section 4.1. The rows of this slot-word matrix contains general slots together with specific slots. In my experiment, the numbers of general slots were 29347 and specific slot phrases 6423 respectively. Therefore, the new slot word table contains $29347 + 6423 = 35770$ slot phrases in all. Note that this shows there are a significant number of specific slot phrases just by using the repeated words in the second position.

   (b) SVD is performed on the slot word matrix with 25 dimensions, to obtain U, S and V.

   (c) The rows of V are the desired low dimensional representation of words in the word space. This word space is the similarity model obtained using specific slot phrases.

3. <u>Comparison</u> This specific-word-space is compared with the general-word-space in the following two steps:

   (a) 10 high frequency words are selected from the 10000 word list. Out of these 5 are randomly selected from the 3000 most frequent words in the 10000 word list. I have selected the other 5 on the basis that these words are a part of a family of objects such as garments, body parts etc.

   (b) For these 10 words, the nearest neighbors are obtained from the word space constructed using general and the specific slot phrases respectively.

   (c) A table listing down the nearest neighbors for each of these 10 words is prepared. The first row of this table contains the nearest neighbors of the word obtained using general slot phrases and the second row contains the neighbors for the specific slot phrases.

For instance, the nearest neighbors of the first word– *meadows*- using specific slot phrases are in the first row – *forests, fields...etc.* Using general slot phrase, the neighbors are, found in the second row – *Missouri,clouds etc.*.

- **<u>Conclusions:</u>** Comparing the nearest neighbors of these 10 words for specific and general slot phrases respectively, the following can be concluded:

  1. Using specific slot phrase have resulted in word space with **much more** semantic similarity than by using general slot phrases. It can be clearly observed that the 10 nearest neighbors obtained using specific slot phrases are semantically very similar to the words.

     For instance, comparing the results for the word *jacket* it can be observed that the nearest neighbors for this word using specific slot phrases are semantically similar words such as *sleeves,waistcoat, shirt, gown, pants etc.* .

     On the other hand the nearest neighbors for *jacket* using general slot phrases are *pouch, profession, hidingplace*, which are not semantically close to *jacket*, on comparison. Similar analysis with each of the other 9 words strongly supports the above conclusion that using specific slot phrases results in very good semantic similarity for high frequency words.
  2. The word space resulting from the use of specific slot phrases has also captured *syntactic similarity.* While being semantically similar, the nearest neighbors for the specific slot phrases belong to the same part of speech category.

- **<u>Discussion</u>** This experiment demonstrates that using specific slot phrases it becomes possible to capture semantic similarity in addition to syntactic similarity for high frequency words. The results are so encouraging that with some more coding, resulting in a spectrum of slot phrases-from very general to very specific-, it would be possible to construct a word space with very accurate semantic information. It

Words	Nearest Neighbours for the Word
meadows	forests, fields, highlands, woodland, canals, spaces, trenches
S	lakes, clouds, chateau, squares, chimneys, interior, roof,cottages
meadows	missouri, clouds, seas, saviour, artillery, roofs, newspapers
G	gates, firstborn, ambassadors, bushes, juice, citizens, grasses, dice
cheerful	happy, jolly, friendly, agreeable, witty, active, pensive
S	hopeless, sorrowful, sickly, miserable, downcast, sickly, pitiful
cheerful	pensive, fertile, disdainful, unreasonable, friendly, hopeless
G	suspicious, wholesome, dangerous, influential, witty, effective
muttered	stammered,murmured, grumbled, mused, chuckled, commented
S	sneered, responded, shouted, growled, yelled, inquired, shuddered
muttered	growled, screamed, shouted, commented, inquired, demanded
G	sighed, bowled, nodded, persisted, insisted, faltered, echoed
spectacles	belt, pillow, coat, gloves, stems, desk, purse, sleeve
S	gown, petticoat, stockings, overcoat, pillows, birthday, medal
spectacles	arrival, throat, ankle, heels, exertions, visage, adversary
G	surroundings, toilet, gloves, backs, pocket, chest, vocation
amazement	astonishment, surprise, rapture,bewilderment,terror
S	excitement, horror, dismay, fright, anger, wrath
amazement	dismay,astonishment, desperation, compliance, anger
G	vanilla,terror, furs, humility, duty, contentment,
chest	waists, ankle, elbows, throat, heels, brow, cheeks
S	backs, chin, forehead, toes, eyelids, lungs, nostrils, helm
chest	pocketbook, belt, toilet, staff, orchard, lungs, chamber, mantle
G	mantle, helm, waist, wand, pouch, rifle, skin, premises, streak
kitchen	sittingroom, bedroom, diningroom,doorway, stairway, staircase
S	fireplace, porch, telephone, curtain, ladder, gateway, stable, parlour, saloon
kitchen	brink, seashore, scaffold, diningroom, capitol,cafe, abyss
G	piano, mainland, churchyard, cathedral, mule, platue, corral
jacket	sleeves, helm, flanks,robe, waistcoat, belt, gown, shirt
S	pants, pullover, umbrella
jacket	pouch, profession, hidingplace, finger, route
G	beak, post, letter, submarine
absurd	ridiculous, idiotic,silly nonsensical,stupid,unexpected,unlikely
S	crazy, weird, illogical
absurd	exciting,goodnatured, delightful, distressing, rudimentary
G	frustrating, brilliant, clever
seventh	sixth,fourth, fifth, eleventh, ninth, twelfth, naval, tenth
S	twentieth, eighth, thirteenth, fifteenth, smallest, youngest
seventh	sixth, eleventh, fifth, twelfth, youngest, biggest, ninth
G	noblest, smallest, muffats, austrian, outer, sixteenth, heathen

Table 4.8: The nearest neighbors resulting from general(G) and specific(S) slot phrases for 10 high frequency words.

would be interesting to construct a hierarchical representation of this word space using hierarchical clustering techniques.

While this experiment has examined the nearest neighbors for 5 random words and 5 typical words, the neighbors for rare words, have not been examined in this experiment.

Nearest neighbors of rare words using specific slot phrases

The previous experiment examined the nearest neighbors of frequent words and it was seen that the nearest neighbors were semantically very similar to the selected words. However, it is not clear whether one would get similar results for rare words. This following experiment examines the nearest neighbors of rare words to see whether they are semantically similar.

- **Aim:** To examine the nearest neighbors of 5 rare words to see whether they are semantically similar.
- **Method:**
  1. The word space generated using specific slot phrases, as described in the previous experiment is taken.
  2. The 500 rarest words from the 10000 word list is listed.
  3. 5 words are chosen at random from this 500 word list.
  4. The 10 nearest neighbors for these 5 words are listed in the table 4.9 and examined to see whether they are semantically similar to the each of the 5 words.
- **Conclusion:**
  1. The nearest neighbors contain very few semantically similar words. The words that are semantically similar are marked with the `typewriter font` in table 4.9.
  2. Some words amongst the nearest neighbors are semantically related. For instance, the nearest neighbors of the word *steamer* are mostly words related to ships and war.
  3. Many words in the nearest neighbors list for words such as *bulk*, *abrupt* and *factors* are not semantically related.
  4. All the nearest neighbors are syntactically very similar.

- **Discussion** The results of this experiment show that the nearest neighbors of rare words are very rarely similar in meaning. However, the words space has captured syntactic similarity. In this sense, the results of this experiment are very similar to those obtained using general slot phrases. Moreover the previous experiement demonstrated that the use of specific slot phrases considerably increased the number of semantically similar words in the nearest neighbors list.

  One possible reason for this result can be the lack of specific slot phrases that tend to relate semantically similar words in the 10000 word list. This problem can be addressed by generating more specific slot phrases, considering the fact that the present technique does not generate all possible specific slot phrases.

  The result can also be explained in terms of the limited size of the corpus presently used. A limited corpus size implies fewer words per slot i.e. words that can *potentially* appear in a slot have not been captured in the corpus. This problem can be solved by using the existing slot phrases and the existing list of 10000 words, and collecting more entries in the slot-word table using a much larger corpus.

  Finally, the nearest neighbors may not be semantically very similar, simply because words very similar in meaning to the given set of words may have not been present in the list in the first place. In the future, I would like to see whether this is indeed the case for a set of rare words.

4.5 Comparison with Schütze's word space

The previous sections described exploratory analysis of the word space obtained using general and specific slot phrase. The following experiment compares the word space with the low dimensional representation of Schütze.

Schütze's word space is interesting because he is able to model syntactic context while using a windows-based approach, as explained in chapter 1. Moreover he uses SVD to obtain a low dimensional word space which captures mostly syntactic and some semantic information, similar to the word space obtained using general slot phrases. Thus, in these respects, Schütze's method seems to be the most similar previously described approach to ours. It would therefore be interesting to see how Schütze's word space compares with the word spaces obtained using our method for general and specific slot phrases respectively.

Words	10 Nearest Neighbours for the Word
hire	`employ`, `remove`, `reject`, decline, `dispose`, undertake refuse, propose, venture, resolve
abrupt	harmonious, `sudden`, continous, ferocious, devilish troublesome, fast, noticable, unmistakable, obscure
bulk	flocks,collections, weapons, industries departments, subjects, opinions, topics, discoveries, interests
factors	chances, meanings, rules,evidences, reports changes, attempts, struggles,glories, triumphs
steamer	`vessel`, barrel, lever, screw, squadron magazine, revolver, `submarine`, fortress, bridge, bomb

Table 4.9: 10 Nearest neighbours for 5 rare words. The `font` is used to mark semantically *very* close words.

I have used two methods in comparing the word spaces: nearest neighbors; and kernel canonical correlation analysis.

The nearest neighbors approach compares the nearest neighbors of 10 selected words from both these spaces. KCCA computes how much the two spaces are correlated. These two experiments are explained in greater detail in the following two sections.

4.5.1 Comparison using nearest neighbors

Examination of the nearest neighbors around a word in a space gives an idea of the information that has been captured by the space. Therefore comparing the nearest neighbors of two spaces can tell us in what way the two spaces are similar or different. The following passages describe this experiment in detail.

.

- **Aim:** To compare the word space obtained using specific slot phrases with that of Schütze using the nearest neighbors of 10 selected words.
- **Method:**
    1. Two word spaces are constructed using slot phrases and Schütze's method respectively.

2. The 10 selected words in the previous experiment(section 4.4.4) are considered and the nearest neighbors from the slot phrase word space and word space using Schütze's technique are listed in table 4.10.
3. The nearest neighbors the 10 words from each of the two spaces are compared to see in what ways they are similar or different from each other.

- **Conclusions:** Comparing the nearest neighbors of the words from either of the two spaces, two conclusions can be drawn:
  1. The nearest neighbors of two spaces are syntactically similar to the words considered. This implies that both these spaces have captured syntactic similarity.
  2. The nearest neighbors obtained using specific slot phrases contain *more* words that are semantically similar than those obtained by Schütze's method.
- **Discussion:** The comparison using nearest neighbors suggests that the Schütze's word space and our space have a lot in common in that they capture syntactic similarity. However, our word space seems to have captured semantic similarity better than Schütze's space. It would be informative to compare our similarity model with other windows based and syntax based models.

4.5.2 Comparison using KCCA

The nearest neighbors approach for comparison of the two word spaces has given an idea of the commonalities and differences between the two spaces for a small sample of words. The following experiment uses kernel canonical correlation analysis to give a more global estimate of how much the two spaces share and how different the two spaces are.

Kernel canonical correlation analysis was introduced in the second chapter as a technique to compare the information conveyed by two spaces. KCCA finds pairs of subspaces from the two spaces which are maximally correlated with each other.

In the following experiment, I have used KCCA to compare Schütze's experiment with the word spaces obtained using general and specific slot phrases respectively. KCCA results in canonical correlations which are compared for the two pairs of spaces. The experiment is described below:

Words	Nearest Neighbours for the Word
meadows Specific SP	forests, fields, highlands, woodland, canals, spaces, trenches lakes, clouds, chateau, squares, chimneys, interior, roof,cottages
meadows Schütze	houses, cups, luxuries, chateau, fields, golf gates, lakes, chimneys, lands, oceans, road, spaces
cheerful Specific SP	happy, jolly, friendly, agreeable, witty, active, pensive hopeless, sorrowful, sickly, miserable, downcast, sickly, pitiful
cheerful Schütze	merciful, bloody, disdainful, friendly, jolly, illustrious furious, drowsy,public,awesome,miserable,agreeable,sickly
muttered Specific SP	stammered,murmured, grumbled, mused, chuckled, commented sneered, responded, shouted, growled, yelled, inquired, shuddered
muttered Schütze	commented, growled, murmured, mused, yelled, sings replied, bothered, shouted, thought, grumbled, responded
spectacles Specific SP	belt, pillow, coat, gloves, stems, desk, purse, sleeve gown, petticoat, stockings, overcoat, pillows, birthday, medal
spectacles Schütze	typewriter, pillow, medal, stems, lotions, birthday desk, highway, petticoat, mouse, papers, stockings
amazement Specific SP	astonishment, surprise, rapture,bewilderment,terror excitement, horror, dismay, fright, anger, wrath
amazement Schütze	dismay, enjoyment, anger, terror, surprise, disappointment imagination, curiosity, amusement, rapture, horror
chest Specific SP	waists, ankle, elbows, throat, heels, brow, cheeks backs, chin, forehead, toes, eyelids, lungs, nostrils, helm
chest Schütze	pocketbook,pen,armor,lungs,throat,belt, toilet,mantle toes, helm, pouch, streak, bones, shoes, drums, chin
kitchen Specific SP	sittingroom, bedroom, diningroom,doorway, stairway, staircase fireplace, porch, telephone, curtain, ladder, gateway, stable, parlour, saloon
kitchen Schütze	seashore,capitol, bedroom, doorway, cafe, dormitory, hostel piano, stars, salad, doorway, curtain, churchyard, clock
jacket Specific SP	sleeves, helm, flanks,robe, waistcoat, belt, gown, shirt pants, pullover, umbrella
jacket Schütze	profession, shirt, pouch, pullover,route, carpet umbrella, gown, paths, mattress, letter
absurd Specific SP	ridiculous, idiotic,silly nonsensical,stupid,unexpected,unlikely crazy, weird, illogical, awkward
absurd Schütze	exciting,distressing,idiotic,weird,awkward,equally brilliant,clever,nonsensical,technical,unlikely,extrodinary
seventh Specific SP	sixth,fourth, fifth, eleventh, ninth, twelfth, naval, tenth twentieth, eighth, thirteenth, fifteenth, smallest, youngest
seventh Schütze	twelfth, biggest, ninth, seventh, eighteenth, naval, seventh smallest, eleventh, tenth, outer, first, youngest, gold

Table 4.10: The nearest neighbors resulting from using Specific Slot phrases(labelled as 'specific SP') and using Schütze's technique(labelled as 'Schütze' for 10 high frequency words.

$No.$	1	2	3	4	5	6	7	8	9	10
Sch-GSP	.82	.80	.71	.69	.67	.64	.62	.56	.50	.37
Sch-SSP	.38	.30	.21	.15	.10	.07	.02	.01	.00	.00

Table 4.11: The first row lists the first 10 canonical correlations obtained from KCCA of Schütze's word space(Sch) and the word space obtained using general slot phrases(GSP). The second row lists the correlations for Schütze's space(Sch) and the word space obtained using specific slot phrases(SSP).

- **Aim:** To compare the information conveyed by two spaces using KCCA.
- **Method:**
    1. The two word spaces– namely that of Schütze and ours is considered for comparison.
    2. KCCA is performed on these two spaces resulting in three matrices $B1$, $B2$ and r. $B1$ and $B2$ are the basis vectors of the two spaces respectively and r is the vector of canonical correlations.
    3. The values of the r vector are observed and are tabulated in table 4.11.

- **Conclusion:** Comparing the two rows of table 4.11, it can be see that the first row has far greater correlations than the second row. Therefore, it can be concluded that Schütze's word space shares greater commonalities with the word space obtained using general slot phrases than with the word space obtained using specific slot phrases.
- **Discussion:** The above conclusion can be explained on the basis that, from the previous experiments, it was seen that both Schütze's word space and the word space obtained using general slot phrases capture more of syntactic rather than semantic similarity. Therefore these spaces would tend to convey similar information as seen from their canonical correlations.

  On the other hand, while the word space obtained using general slot phrases does capture syntactic similarity, it captures semantic similarity in addition. Thus it conveys more information than Schütze's space. This difference is apparent in their canonical correlations tabulated in the second row.

  Using KCCA, it may be possible to understand the nature of commonalities and differences shared by the spaces analyzed in this experiment.

This can be achieved by projecting the two spaces on to the basis vectors and look at the arrangement of the words in the resulting spaces. We would like to explore this possibility in the future.

Chapter 5

Discussion and future work

In this thesis, we have described a new approach to word similarity modelling using a novel definition of a context called “slot phrases”. According to this approach, common patterns of words can act as good contexts for modelling word similarity. Slot phrases capture these “common patterns of words” by a procedure, which automatically extracts “patterns of common words” from the text.

Slot phrases require little linguistic prior knowledge as they are based on the statistical properties of the text corpus. Slot phrases use these properties to model a spectrum of similarity relationships–from general to specific– by using a spectrum of word patterns– from common to rare.

We have demonstrated that this approach performs well in capturing syntactic and semantic similarity. This performance is achieved despite the fact that the methods used to construct slot phrases are very simple and need many extensions to gain greater benefits of this intuitively appealing approach.

Slot phrases can be easily extended in many ways. Firstly, common patterns of words have been identified with the help of punctuation marks such as commas in the present approach. However, common patterns of words can occur across punctuation marks, even across full stops. For example, there may be a common pattern in the way some sentences end and the next sentence begins. Therefore a more statistical approach to find the most common patterns of words can be used instead of the present method.

Secondly, the present technique for extracting specific slot phrases can be easily extended, with more coding, using the same method that has been used in this work. It would then be possible to model a spectrum of similarity relationships– from very general relationships to specific ones.

While the word space constructed using general slot phrases have been more extensively analysed in this work, the one constructed using specific slot phrases needs to be analysed further. Using a range– from general to specific– slot phrases may lead to a space which may capture very interesting properties of similarities between words. Hiearchical and other types of clustering such as spectral clustering on this word space would be a very interesting extension to the present work.

Finally, more experiements can be carried out to understand why the present word space is not able to capture much of semantic similarity for the rarer words in the corpus. Using a much larger corpus it would be possible to have many more *specific slot entries* for every word. At the same time, a much larger set of words can be used for the word space, ensuring that for many words, the semantically similar words are present in the list. Much better results for semantic similarity may be expected by relating a larger proportion of these words using specific slot phrases.

In addition to these extensions discussed above, the present work opens up interesting directions to explore using slot phrases. One possiblity is to allow more than one word to appear in a slot, so that it may become possible to model similarity relationship between words and phrases, and similarity between different phrases. Another interesting direction is to extend the present paradigm of using common patterns of words for similarity modelling to other languages, including langauges which have more free word order than English.

This thesis is something of a pilot project,a first attempt at defining a novel definition of a context using common patterns of words in text. This study would be more complete after exploring the above extensions to the present work and also after an extensive comparison to other similarity models. Meanwhile, we hope that this thesis has demonstrated the potential of the slot-phrase based approach and has opened up new possiblities for word similarity modelling.

Bibliography

[1] J.R. Curran and M. Moens. Improvements in automatic thesaurus extraction. *Proceedings of the Workshop on Unsupervised Lexical Acquisition*, pages 59–67, 2002.

[2] Lillian Lee Fernando Pereira, Naftali Tishby. Distributional clustering of english words. *In proc. of the Annual Meeting of the ACL*, 1993.

[3] Caroline Gasperin, Pablo Gamallo, Alexandre Agustini, Gabriel Lopes, and Vera de Lima. Using syntactic contexts for measuring word similarity. *Text, Speech and Discourse(TSD-2001)*, pages 116–125, 2001.

[4] G.Grefenstette. Sextant: exploring unexplored contexts for semantic extraction from syntactic analysis. *Proceedings of the 30st annual meeting of the Association for Computational Linguistics, ACL*, 1992.

[5] Derrick Higgins. Which statistics reflect semantics? rethinking synonymy and word similarity. *conference volume from the 2004 International Conference on Linguistic Evidence*, 2004.

[6] Donald Hindle. Noun classification from predicate-argument structures. *In 28'th Annual Meeting of Association for Computational Linguistics*, pages 268–275, 1990.

[7] Fernando C.N.Pereira Ido Dagan, Lillian Lee. Similarity-based models of word-cooccurence probablities. *Kluwer Academic Publishers*, pages 1–31, 1997.

[8] Shaul Markovitch Ido Dagan, Shaul Marcus. Contextual word similarity and estimation from sparse data. *Computer Speech and Language*, 9:123–152, 1995.

[9] Kanerva P. Kristofersson J. and Holst A. Random indexing of text for latent semantic analysis. *Procedings of the 22nd Annual Conference of the Cognitive Science Society*, page 1036, 2000.

[10] Dekang Lin. Automatic retrieval and clustering of similar words. *In Proceedings of COLING-ACL98*, 58:768–774, 1998.

[11] K. Lund and C. Burgess. Producing high-dimensional semantic spaces from lexical co-occurance. *Behavior Research Methods, Instruments and Computers,28(2)*, pages 203–208, 1996.

[12] H.D.Sherali M.S.Bazaraa and C.M.Shetty. Nonlinear programming: theory and algorithms. *Wiley, second edition*, 1993.

[13] C.-J. Lin P.H. Chen and B. Scholkopf. A tutorial on nu -support vector machines.

[14] Magnus Sahlgren. Vector-based semantic analysis: Representing word meanings based on random labels. *Semantic Knowledge Acquisition and Categorisation Workshop, ESSLLI '01, Helsinki, Finland*, 2000.

[15] Hinrich Schutze. Dimensions of meaning. *Proceedings of Supercomputing*, 1992.

[16] Hinrich Schutze. Part of speech induction from scratch. 1993.

[17] Hinrich Schutze and J.O Pederson. A co-ocurance-based thesaurus and two applications to information retrieval. *Information Processing and Management33(3)*, pages 307–318, 1997.

[18] Jody S.Hourigan and Lynn V.McIndoo. The singular value decomposition.

[19] John Shawe Taylor. An introduction to support vector machines. *Cambridge University Press.*

[20] John Shawe Taylor. Kernel methods for pattern analysis. pages 141–142, 2004.

[21] John Shawe Taylor. Kernel methods for pattern analysis. pages 164–175, 2004.

[22] Darrel Laham Thomas.K.Landauer and Peter Foltz. Learning human like knowledge using singular value decomposition. *Advances in Neural Information Processing System 10*, pages 45–51, 1998.

[23] Vladimir Vapnik. The nature of statistical learning theory. *Springer-Verlag, Newyork*, 1995.