UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of master’s thesis by

AMRUTA PURANDARE

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Dr. Ted Pedersen

Name of Faculty Adviser

Signature of Faculty Advisor

Date

GRADUATE SCHOOL

Unsupervised Word Sense Discrimination by Clustering Similar Contexts

A THESIS

SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL

OF THE UNIVERSITY OF MINNESOTA

BY

Amruta Purandare

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

August 2004

Contents

List of Figures

List of Tables

Abstract

Word sense discrimination is the problem of identifying different contexts that refer to the same meaning of an ambiguous word. For example, given multiple contexts that include the word ’sharp’, we would hope to discriminate between those that refer to an intellectual sharpness versus those that refer to a cutting sharpness. Our methodology is based on the strong contextual hypothesis of Miller and Charles (1991), which states that ”two words are semantically related to the extent that their contextual representations are similar.”

This thesis presents corpus–based unsupervised solutions that automatically group together contextually similar instances of a word as observed in a raw text. We do not utilize any manually created or maintained knowledge–rich resources such as dictionaries, thesauri or annotated corpora. As a result, our approach is well suited to the fluid and dynamic nature of word meanings. It is also portable to different domains and languages, and scales easily to larger samples of text.

The overall objective of this thesis is to study the effect of various feature types, context representations and clustering methods on the accuracy of sense discrimination. We also apply dimensionality reduction techniques to capture conceptual similarities among the contexts and don’t just rely on the surface forms of words in the text. We present a systematic comparison of various discrimination techniques proposed by Pedersen and Bruce (1997) and Schutze (1998). We find that the first order method of Pedersen and Bruce performs well with larger amounts of text, but that the second order method of Schutze is more effective with smaller data sets. We also discovered that a divisive approach is more suitable for clustering smaller set of contexts, while the agglomerative method performs better on larger data. We conducted experiments to study the effect of using various sources of training, and found that local contexts of a word provide better discrimination features than a running text like complete newspaper articles. We compared the performance of our knowledge–lean method against that of a knowledge–intense approach, and found that although the latter was successful in conjunction with smaller datasets, it didn’t show significant improvements with larger data. This suggests that the features learned from a large sample of text certainly have the potential to outperform those learned from a knowledge-rich resource like dictionary.

1 Introduction

Most words in natural language have multiple possible meanings. The intended meaning of an ambiguous word can be determined by considering the context in which it is used. Given an ambiguous word used in a number of different contexts, *word sense discrimination* is the process of identifying which of those contexts refer to the same meaning of that word. This ambiguous word under consideration is often referred to as the *target word*.

When we observe a target word used in some written text, we call it an *instance* of that word. The term *context* is used to refer to 2 or 3 sentences that around an instance of the target word. For example, if the target word is *Shells*, then word sense discrimination tries to distinguish among the instances of *Shells* that refer to *Sea Shore Shells* versus those that refer to *Bomb Shells* or *Unix Shells*.

Approaches to this problem are often based on the strong contextual hypothesis of Miller and Charles [25], which states that : *two words are semantically related to the extent that their contextual representations are similar*. Hence the problem of word sense discrimination reduces to that of determining which instances of a given target word are used in similar contexts.

In this thesis, we take a corpus–based machine learning approach to achieve sense discrimination. Our algorithm first learns a set of common word patterns observed in the context of a target word in a large sample of text, and then discriminates given instances using clustering algorithms that automatically group together the instances using similar patterns in their contexts. The word patterns selected for making such distinctions are referred to as *features*. Thus, the output of a sense discrimination system shows clusters of given text instances such that the instances grouped in the same cluster are contextually more similar to each other than they are to the instances grouped in the other clusters. As the instances in the same cluster use the target word in similar contexts, we can presume that they all refer to the same meaning of that word. Thus, each cluster is supposed to represent a single word meaning, which is used by all instances grouped in that cluster.

Some may wonder about questions like: *What if the contexts referring to the same meaning do not use same words?* or *What if the contexts referring to different meanings are using the same words?* These are typical challenges faced when dealing with automatic approaches to natural language processing, and fortunately, there is a solid body of research upon which to draw for solutions. The first problem is due to *synonymy*, which means there are many different words people can use to refer to the same underlying concept. For example, consider the following two sentences:

> *Apple unveiled a new family of wide–screen flat panel displays.*
> *Apple released their largest high resolution screen ever.*

These statements announce the same news without using any word in common, except *Apple* which is acting as the target word. The next question that might be raised is: *How do you automatically determine that 'displays' and 'screen' refer to the same thing?* or *Could you provide an online thesaurus to look for synonyms, or a dictionary to look for meanings of these words?*

Our belief is that, in general, any approach to analyzing the meaning of contexts that depends on manually created and maintained resources will fail. Information in the real world is dynamic, with the best example being the World Wide Web.

Every day approximately 1.5 million pages are added to the Web. This introduces new terminology and word usages to refer to new personalities, companies, business products, and phenomena. By contrast, manually written dictionaries and thesauri are relatively stagnant and undergo changes very slowly over a period of years or even decades. Such resources, while being of very high quality, can not cope with the rapidly changing vocabulary of this dynamic world.

For example, most dictionaries do not include the newer sense of the word *apple*, as in *Apple Computers*, which is in fact the most frequent sense of *apple* on the Web or in the news media.

The objective of this thesis is to develop a highly portable and easily adaptable methodology that learns word meanings automatically from raw text. Thus, instead of using information from a dictionary or thesaurus, we refer to an available corpus of electronic text, and then automatically identify which words tend to occur together very often. According to the strong contextual hypothesis, words observed in similar contexts are semantically related. For example, our approach will automatically determine that the words *display* and *screen* often co-occur with each other or co-occur with other similar words like *monitor, resolution, color, vision, pixels,* etc. and hence are related.

While there has been some previous work in word sense discrimination (e.g. [38], [30], [31], [39], [13]), by comparison it is much less than that devoted to word sense disambiguation. Disambiguation is distinct from discrimination in at least two respects. First, the number of possible senses a target word may have is usually not known in discrimination, while disambiguation is often viewed as a classification problem where an instance of the target word is assigned to one of its possible senses that are pre–defined. Second, discrimination can be achieved using no knowledge outside raw text, whereas approaches to disambiguation often rely on supervised learning in which a system learns from manually created examples that show the intended sense of the target word in various contexts. Text in which the instances of a target word are manually tagged with their correct sense is referred to as sense–tagged text. The creation of sense–tagged text is time consuming and results in a knowledge acquisition bottleneck that severely limits the portability and scalability of systems that employ it. By contrast, word sense discrimination can be achieved using purely knowledge–lean unsupervised techniques that do not rely on any knowledge intensive resources like sense–tagged text or dictionaries. Contexts are clustered based on their mutual similarities which are completely computed from the text itself.

While this thesis mainly addresses the problem of word sense discrimination, the techniques we discuss here essentially apply to any task that requires clustering of similar units of text, ranging from single sentences, to paragraphs, to entire documents. For example, one might be interested in automatically organizing their personal emails or files into folders, or might wish to categorize news articles collected from various online news resources according to the topic of news. One can also create clusters of related words (those used in similar contexts) to automatically build a thesaurus or an ontology. In short, the topic of discriminating text units based on their contextual (and hence conceptual) similarities targets a broader range of applications from information retrieval, document clustering/indexing, text categorization, synonymy identification, automatic ontology acquisition and so on.

The various contributions of this thesis to research on word sense discrimination are briefly summarized below, and will be discussed in more detail throughout the thesis.

1. We compared the discrimination techniques proposed by Pedersen and Bruce [30], [31] and by Schütze [38], [39]. We observed that there are some significant differences in their approaches and as yet there has not been any systematic study to determine which results into better discrimination. This thesis tries to address this question via an extensive experimental analysis.

2. We varied our discrimination experiments with respect to various parameters such as feature types,

context representations, and clustering methods to determine which combination of settings resulted in the most accurate results. The overall objective of this thesis is to see if any particular combination of these parameters discriminates best under all/certain conditions.

3. We observed that the nature and volume of data used for feature selection and clustering is a critical factor in the performance of discrimination. In particular, our experiments confirmed that the quality of features (and hence discrimination) improves considerably with increased amounts of data used for feature selection.

4. When only small amounts of data are available for feature selection, we observed that first order context representations that only record the information about features that actually appear in the context do not prove very effective. This is partially due to the inherent sparsity in natural language text combined with a smaller feature set used to represent the contexts. In such case, we realized that the technique of incorporating additional information about feature words into contexts (as used by the second order context representations) considerably improves the results.

5. We observed that, sparse context representations using a small set of features tend to have very low similarities among most pairs of contexts. In such case, the agglomerative clustering method that rigorously compares similarities among the contexts in a pairwise fashion doesn't discriminate accurately. On the other hand, a divisive approach to clustering as taken by the Repeated Bisections method seems to perform better.

6. Larger amounts of data results in better features, which in turn allows for direct comparisons among the contexts from their first order representations. We noticed that with the better quality of features, additional information as included by the second order contexts is not necessary or in fact deteriorates the performance by obscuring distinctions among the contexts referring to different senses.

7. With a sufficient amount of data, the successive comparisons done by the agglomerative clustering method prove more effective than the hybrid partitional approach taken by the Repeated Bisections method.

8. We conducted experiments by selecting features from two types of datasets; local data which is simply a collection of contexts around a specific target word, and global data like newspaper text where a

target word may not appear in every context. Our results showed that the global data though was used in a very large quantity didn't prove to be as useful as the smaller amount of local data.

9. We compared the results of our knowledge–lean approach against those obtained with a more knowledge–intensive method that incorporated actual dictionary meanings of feature words into contexts. We observed that this knowledge–intensive technique only proved more accurate than the experiments conducted with smaller data. This confirmed our hypothesis that features learned from a large text have the potential to outperform those learned from a knowledge rich resource like a dictionary.

10. We have developed an open source software package called SenseClusters that is freely distributed under the GNU Public License. All experiments reported in this thesis can be re-created using the programs and scripts provided in this package. The interested reader is encouraged to download and examine the package from `http://senseclusters.sourceforge.net`.

2 Background

Clustering methods divide a given set of objects into some number of meaningful clusters, where objects grouped into the same cluster are more similar to each other than they are to objects in other clusters.

Clustering is distinct from classification, in that the latter is the problem of assigning an object to one of a pre–defined set of categories. Clustering uses a data–driven approach in which objects are grouped purely based on their mutual similarities without any knowledge of existing classes [16] [17].

The problem of clustering can be divided into the following steps:

1. Feature Selection: identify significant attributes of objects that help to make distinctions between various natural groupings.

2. Object Representation: convert objects to a form that is easy to process by the clustering algorithm.

3. Clustering: mutual similarities between objects are computed, and they are clustered based on these values.

4. Evaluation: the resulting clusters can be compared relative to an existing clustering that is known to be correct.

In the following sections we will describe each step in more detail.

We will refer to the problem of document clustering to illustrate some of the key concepts. In this problem, a set of documents is analyzed, and those documents that are about the same or a similar topic should be clustered together.

2.1 Feature Selection

Features are the distinguishing attributes of objects that help to discriminate among the objects. The choice of features is crucial because carefully chosen informative features improve discrimination among objects, and poorly chosen noisy features can confuse the clustering process. For example, in document clustering, one might use the most frequently occurring words or the words in the title of the document as features.

Though there is no single recommended strategy for feature selection that applies to all clustering problems, there are some heuristics that can be employed that will avoid obviously bad features:

1. Features that are common to all objects can be omitted. For instance, if *computers* occurs in all documents, it doesn't help to distinguish among the different topics present in the documents.
2. Features that are attributes of only single object can also be avoided. This is because a clustering algorithm looks for similarities among the objects, and an attribute characterized by a single object will not be shared by any other object in the given collection. For example, if *psychology* occurs in only one document, it can be eliminated from the feature set.

These heuristics suggest that we put some lower and upper frequency bounds on features. As such, we specify the minimum and maximum number of objects that should exhibit a certain feature in order for it be included in our feature set.

In document clustering, an upper limit on the number of times a word occurs will automatically exclude many high frequency (low information content) words like *the, is, are, of,* and to. In addition, very rare words can also be excluded, since they provide a level of detail that is too fine grained for making topic distinctions.

2.2 Object Representation

Once the set of features is selected, the value of each feature is measured for each object. Features may be numeric or strings.

In the case of binary features, the value is 1 if the feature occurs, and 0 if it does not. These are typically used for features that represent whether or not a particular word occurs in a document. Numeric features may also have integer or real values. For example, an integer feature could record the number of times a particular word occurs in a document.

String valued features can be very descriptive. For example, suppose we have a feature that indicates the origin of a document ('document_source'). It might have possible values such as *newspaper, journal, book, web,* and *conference-proceedings*, all of which describe where a document originally appeared.

For string valued features, it is common practice to assign numeric identifiers to these features, in the interests of computational convenience and to reduce storage requirements. So rather than storing and manipulating these strings, we identify them by numeric values, such that *newspaper* becomes 1, *journal* becomes 2, *book* becomes 3, and so forth.

Real valued features are useful when making more precise measurements than integers or binary features allow. For example, suppose instead of using single word features, we now have features that represent two word sequences (bigrams) that occur in computer related documents such as *software engineering, information technology, operating system, computer architecture,* or *network security*. A real valued feature can represent the scores of measures of association such as the log–likelihood ratio or mutual information.

Selected features can be viewed as the dimensions of a multi-dimensional space in which given objects can be represented either as vectors or as points. The feature values then define the values of vector components or point co-ordinates.

Consider a simple 2-D space formed by the features *computers* and *finance*. Suppose in document1 that we observe feature *computers* 3 times and *finance* once (in other words, the value of these features is 3 and 1, respectively). Then suppose in document2 that *computers* occurs twice and *finance* occurs 4 times. Given this scenario, we can view document1 as a vector $(3i + 1j)$ and document2 as $(2i + 4j)$, which is shown in Figure 1. In co-ordinate space, document1 can be seen as the point $(3, 1)$, while document2 as the point $(2, 4)$, as shown in Figure 2.

When features are binary, objects can be represented as sets or unordered lists of features that are attributes of that object. For example, if a document includes the terms *firewall, security, recovery, virus, authentication,* and *encryption*, then the set representation of this document will be (*firewall, security, recovery, virus, authentication, encryption*). Such a feature set can also be viewed as a sparse binary vector in the feature space that is formed by the union of all the object's feature sets.

2.3 Measuring Similarities

The objects to be clustered can be represented as vectors, points or sets in the feature space. The next step is to compute their mutual similarities. There are a variety of well known measures that can be employed, and we briefly review them below.

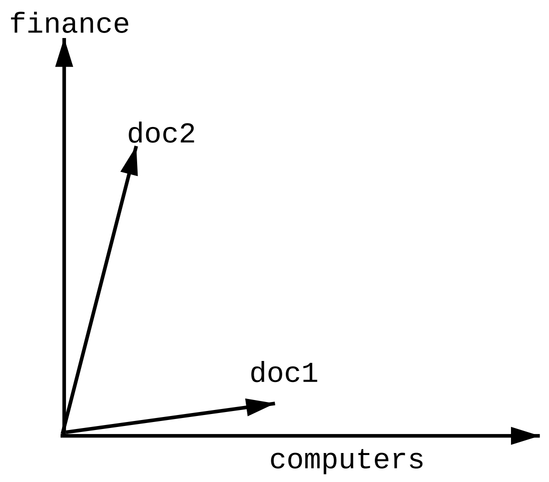

Figure 1: Vector Representations of Objects

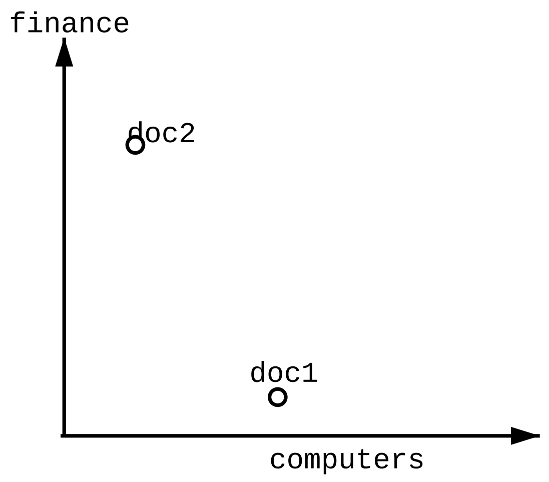

Figure 2: Point Representations of Objects

2.3.1 Real-Valued Feature Space

These measures are employed with real–valued feature spaces, and can also be used with integer or binary feature spaces.

Cosine Similarity Coefficient This measure requires that objects be represented as vectors, and measures their similarity by taking the cosine of the angle between two vectors:

$$COS(\overrightarrow{P},\overrightarrow{Q}) = \frac{\overrightarrow{P} \cdot \overrightarrow{Q}}{\mid \overrightarrow{P} \mid\mid \overrightarrow{Q} \mid} \tag{1}$$

where $\overrightarrow{P}$ and $\overrightarrow{Q}$ are each feature vectors associated with a particular object.

Objects that have similar feature values will be closer in the space and have a smaller angle between their vectors, which results in a higher cosine value. If the two vectors are identical, cosine is 1. On the other hand, if the vectors do not share any features then the cosine is 0.

Object vectors are often normalized so that a vector with a large scale does not dominate the other vectors:

$$\mid \overrightarrow{v} \mid = \sqrt{\sum_{i=1}^{N} v_i^2} \tag{2}$$

$$normalized(\overrightarrow{v}) = \frac{\overrightarrow{v}}{\mid \overrightarrow{v} \mid} \tag{3}$$

The first equation above shows the norm of the vector *v*.

If vectors P and Q in equation 1 are normalized as shown by equation 3 then

$$\mid \overrightarrow{P} \mid = \mid \overrightarrow{Q} \mid = 1. \tag{4}$$

Then, the cosine of the angle between the two vectors reduces to their dot product:

$$COS(\overrightarrow{P},\overrightarrow{Q}) = \overrightarrow{P} \cdot \overrightarrow{Q} \tag{5}$$

Example If

$$\vec{v1} = (0.4, 0, 0.07, 0.348, 0) \tag{6}$$

and

$$\vec{v2} = (0.32, 0.1, 0, 0.593, 0.2) \tag{7}$$

then

$$\begin{aligned} & COS(\vec{v1}, \vec{v2}) \\ &= \frac{0.4 \times 0.32 + 0 \times 0.1 + 0.07 \times 0 + 0.348 \times 0.593 + 0 \times 0.2}{\sqrt{0.4^2 + 0 + 0.07^2 + 0.348^2 + 0} \times \sqrt{0.32^2 + 0.1^2 + 0 + 0.593^2 + 0.2^2}} \\ &= \frac{0.128 + 0 + 0 + 0.206364 + 0}{\sqrt{0.286004} \times \sqrt{0.504049}} \\ &= \frac{0.334364}{0.534793 \times 0.709964} \\ &= 0.880638 \end{aligned}$$

Euclidean Distance This measure computes the spatial or straight line distance between two points in a N-Dimensional space:

$$d = Dist(P, Q) = \sqrt{\sum_{i=1}^{N} (P_i - Q_i)^2} \tag{8}$$

Then similarity between objects can be expressed in terms of their distance [17]:

$$Sim(P, Q) = \frac{1}{1 + d} \tag{9}$$

If the objects are exactly identical, distance between them will be 0 and similarity will be 1. Distance is not normalized and hence distance increases as the objects move farther from each other.

Example If P=(0.4, 0, 0.07, 0.348, 0) and Q=(0.32, 0.1, 0, 0.593, 0.2), then

$$\begin{aligned} Dist(P, Q) &= \sqrt{(0.4 - 0.32)^2 + (0 - 0.1)^2 + (0.07 - 0)^2 + (0.348 - 0.593)^2 + (0 - 0.2)^2} \\ &= \sqrt{0.0064 + 0.01 + 0.0049 + 0.060025 + 0.04} \end{aligned}$$

$$= \sqrt{0.121325}$$

$$= 0.348317$$

2.3.2 Binary-Valued Feature Space

These measures operate in binary-valued feature space and assume that the objects to be clustered are represented as sets.

Match Coefficient This takes the set intersection of a pair of feature sets and indicates how many features are shared by the two sets. In short, it is the cardinality of the intersection of the two feature sets.

$$Match(P, Q) = |P \cap Q| \tag{10}$$

Example If P={film, story, actor, photography, theater, picture, stage} and Q={story, theater, perform, actor}, then,

$$Match(P, Q) = |P \cap Q| = |\{story, actor, theatre\}| = 3 \tag{11}$$

Dice Coefficient This divides the cardinality of the intersection of the two sets by the sum of their lengths. This measure is normalized to the $[0, 1]$ scale by multiplying by 2.

This measure attempts to account for the size of the sets being compared, in addition to simply determining how many features match:

$$Dice(P, Q) = \frac{2 \times |P \cap Q|}{|P| + |Q|} \tag{12}$$

Example If P={film, story, actor, photography, theater, picture, stage} and Q={story, theater, perform, actor}, then,

$$\begin{aligned} Dice(P, Q) &= \frac{2 \times |P \cap Q|}{|P| \times |Q|} \\ &= \frac{2 \times 3}{7 + 4} \end{aligned}$$

$$= \frac{6}{11}$$
$$= 0.5455$$

Jaccard Coefficient This divides the cardinality of the intersection of the two sets by the cardinality of their union. The objective of this measure is to give lower similarity scores to long sets that have smaller intersections.

$$Jaccard(P, Q) = \frac{|P \cap Q|}{|P \cup Q|} \tag{13}$$

Example If P={film, story, actor, photography, theater, picture, stage} and Q={story, theater, perform, actor}, then,

$$\begin{aligned} Jaccard(P, Q) &= \frac{|P \cap Q|}{|P \cup Q|} \\ &= \frac{3}{8} \\ &= 0.375 \end{aligned}$$

Overlap Measure This divides the cardinality of the intersection by the minimum of the two set lengths. This measure accounts for the case when one of the sets is smaller than the other, and gives a higher score to smaller overlaps in such a case. The maximum value of this measure is 1, and this value is reached if one of the sets is a subset of the other.

$$Overlap(P, Q) = \frac{|P \cap Q|}{min(|P|, |Q|)} \tag{14}$$

Example If P={film, story, actor, photography, theater, picture, stage} and Q={story, theater, perform, actor}, then,

$$\begin{aligned} Overlap(P, Q) &= \frac{|P \cap Q|}{min(|P|, |Q|)} \\ &= \frac{3}{min(7, 4)} \end{aligned}$$

$$= \frac{3}{4}$$
$$= 0.75$$

2.3.3 Similarity Matrix

The pairwise similarities between N objects are often represented in a N x N *similarity matrix*. The rows and columns of such a matrix represent the objects, and the cell value at *(i,j)* indicates the similarity between the pair of the objects at the corresponding indices.

Table 1 shows an example of a matrix representing pairwise similarities between 5 documents. Note that a similarity matrix is always symmetric because the similarity of pair (i,j) is same as the similarity of pair (j,i).

	D1	D2	D3	D4	D5
D1	1	0	0	0.21	0.3
D2	0	1	0.63	0.42	0
D3	0	0.63	1	0.35	0.2
D4	0.21	0.42	0.35	1	0
D5	0.3	0	0.2	0	1

Table 1: Similarity Matrix for 5 Document Example

This matrix can also be viewed as a graph (Figure 3) with N vertices and $\frac{NNZ}{2}$ edges, where NNZ=Total number of non-zero values in the similarity matrix. The vertices of this graph represent the given objects while the edges connect pairs that have non-zero similarity values. This results in a space efficient sparse representation, since any pairs with a similarity of 0 do not have an edge connecting them.

2.4 Dimensionality Reduction via SVD

Suppose we have a collection of 8 documents taken from computer and medical journals. Table 2 shows a document by term matrix in which rows represent the documents and columns represent the terms (i.e., words or word sequences) that occur in these documents.

Each cell entry at (i, j) indicates the frequency of the term represented by the j^{th} column in the document

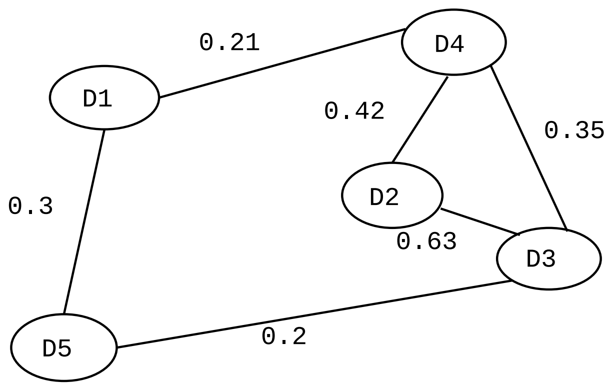

Figure 3: Graphical Representation of Objects

	apple	blood	cells	ibm	data	desktop	tissue	graphics	memory	organ	plasma
C1	2	0	0	1	3	1	0	0	0	0	0
M1	0	3	0	0	0	0	2	0	0	2	1
C2	1	0	0	2	0	3	0	1	2	0	0
M2	0	2	1	0	0	0	2	0	1	0	1
M3	0	0	3	0	2	0	2	0	2	1	3
C3	0	0	0	2	3	0	0	1	2	0	0
C4	2	0	0	1	3	2	0	1	1	0	0
C5	0	0	0	2	3	4	0	2	0	0	0

Table 2: Document by Term Association Matrix

represented by the i^{th} row. Note that the names of medical documents start with M while those of the computer documents start with C. The rows of the matrix can be viewed as the vectors or point co-ordinates that represent the corresponding documents in the 11-Dimensional space formed by the selected term features.

One limitation of this representation is that it fails to address the problems of polysemy (a single term with multiple meanings) and synonymy (multiple terms having the same meaning) inherent in natural languages.

Notice in table 2 that, ambiguous words like apple, tissue, organ are represented with a single dimension. If the similarities between the documents are computed by literally matching their features, this can result in false or mistaken matching. This might cause unrelated documents to receive artificially high similarity scores, and hence be grouped together. Instead, we want to distinguish between the documents that use the same terms but with different meanings.

Another limitation of this object representation is that it gives synonymous features separate dimensions. For example, the words *blood* and *plasma*, as shown above. In practice, we may not want to make fine distinctions in their meanings and would want to call the documents similar even if one uses the term *blood* and other uses *plasma*. In other words, we want to recognize the use of synonyms, so that we do not artificially distance two closely related documents that happen to choose different words to represent the same meaning.

Fortunately, Latent Semantic Indexing (LSI) [4] [3] and Latent Semantic Analysis (LSA) [10] [19] address both polysemy and synonymy. Specifically, LSI/LSA use a dimensionality reduction technique called Singular Value Decomposition (SVD) [2] that causes dimensions associated with synonyms to come together, and differentiates between the various meanings of a polysemous word.

SVD decomposes any rectangular (m x n) matrix into the product of 3 matrices:

$$SVD(A) = UDV' \tag{15}$$

where, matrices U and V contain the left and right singular vectors of A and D is a matrix of singular values of A [40].

Properties of U, D and V U and V represent the orthonormal basis for the column and row span of A, which means,

1. The columns of U and V are orthogonal. All columns of U (and V) are linearly independent and the dot product of any two columns is 0.
2. The norm of each column of U (and V) is 1.
3. U and V form the basis for span of row and column space of A, meaning all columns of U and V are linearly independent and all columns(rows) of A can be represented by some linear combinations of columns(rows) of U(V).

0.35	0.09	-0.20	0.52	-0.09	0.40	0.02	0.63	0.20	-0.00	-0.02
0.05	-0.49	0.59	0.44	0.08	-0.09	-0.44	-0.04	-0.60	-0.02	-0.01
0.35	0.13	0.39	-0.60	0.31	0.41	-0.22	0.20	-0.39	0.00	0.03
0.08	-0.45	0.25	-0.02	0.17	0.09	0.83	0.05	-0.26	-0.01	0.00
0.29	-0.68	-0.45	-0.34	-0.31	0.02	-0.21	0.01	0.43	-0.02	-0.07
0.37	-0.01	-0.31	0.09	0.72	-0.48	-0.04	0.03	0.31	-0.00	0.08
0.46	0.11	-0.08	0.24	-0.01	0.39	0.05	-0.75	0.08	-0.00	-0.01
0.56	0.25	0.30	-0.07	-0.49	-0.52	0.14	0.07	-0.30	0.00	-0.07

Table 3: Matrix U

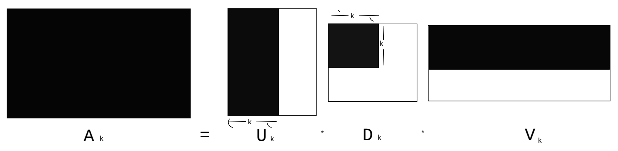

Figure 4: Reducing a Matrix to K Dimensions with SVD

The number of columns in U and V are referred to as the dimensionality of the column and row space of A.

D is a diagonal matrix where all entries except the diagonal are zeros. The diagonal values of D are called singular values which show the significance of each dimension in the corresponding column and row space of A. For further computational ease, diagonal values of D are arranged in the descending order.

Tables 3, 4 and 5 show the matrices U, D and V obtained after performing SVD on matrix A (table 2).

When multiplying matrices U, D and V', the original matrix A is returned. However, the goal of LSI/LSA is not to simply recover the original matrix, but rather to get a reduced matrix that contains much the same information, but represented in fewer dimensions (say k). This is achieved by selecting first k significant singular values from matrix D or by setting all its diagonal entries beyond k+1 to 0s. This has the effect of reducing the dimensionality of matrix A to k dimensions.

Figure 4 shows how matrices U, D and V' are truncated to obtain the best approximation of matrix A (A_k) in k dimensions where k $<$ rank(A).

9.19	0	0	0	0	0	0	0	0	0	0
0	6.36	0	0	0	0	0	0	0	0	0
0	0	3.99	0	0	0	0	0	0	0	0
0	0	0	3.25	0	0	0	0	0	0	0
0	0	0	0	2.52	0	0	0	0	0	0
0	0	0	0	0	2.30	0	0	0	0	0
0	0	0	0	0	0	1.26	0	0	0	0
0	0	0	0	0	0	0	0.66	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

Table 4: Matrix D

0.21	0.08	-0.04	0.28	0.04	0.86	-0.05	-0.05	-0.31	-0.12	0.03
0.04	-0.37	0.57	0.39	0.23	-0.04	0.26	-0.02	0.03	0.25	0.44
0.11	-0.39	-0.27	-0.32	-0.30	0.06	0.17	0.15	-0.41	0.58	0.07
0.37	0.15	0.12	-0.12	0.39	-0.17	-0.13	0.71	-0.31	-0.12	0.03
0.63	-0.01	-0.45	0.52	-0.09	-0.26	0.08	-0.06	0.21	0.08	-0.02
0.49	0.27	0.50	-0.32	-0.45	0.13	0.02	-0.01	0.31	0.12	-0.03
0.09	-0.51	0.20	0.05	-0.05	0.02	0.29	0.08	-0.04	-0.31	-0.71
0.25	0.11	0.15	-0.12	0.02	-0.32	0.05	-0.59	-0.62	-0.23	0.07
0.28	-0.23	-0.14	-0.45	0.64	0.17	-0.04	-0.32	0.31	0.12	-0.03
0.04	-0.26	0.19	0.17	-0.06	-0.07	-0.87	-0.10	-0.07	0.22	-0.20
0.11	-0.47	-0.12	-0.18	-0.27	0.03	-0.18	0.09	0.12	-0.58	0.50

Table 5: Matrix V

0.05	-0.49
0.35	0.13
0.08	-0.45
0.29	-0.68
0.37	-0.01
0.46	0.11
0.56	0.25

Table 6: Truncated Matrix U

9.19	0
0	6.36

Table 7: Truncated Matrix D

Tables 6, 7 and 8 show the matrices U, D and V truncated after retaining the top two dimensions. On multiplying the truncated forms of U, D and V', we get A_k as shown in table 9.

Note that most of the zeros in the original matrix (table 2) are now smoothed to some non-zero values. Thus, SVD has a smoothing effect which assigns some non-zero values to features that do not actually occur in the documents but occur in similar documents. In Table 2, the term *blood* does not occur in the document M3 but after SVD (see table 9), this term gets a higher score (1.7). This is because document M3 contains medical terms like *plasma* and *cells*, and all other documents that contain these terms also include *blood*. The same is true for the term *apple* that doesn't actually appear in C5 but still gets a high score of 1.22 after SVD. This shows how SVD can improve the similarity score between texts that may use different terminology for the same concepts.

The reduced matrix no longer represents the actual words that occur in a text, but rather dimensions that suggest underlying concepts. This has the effect of converting a word level feature space into a concept level semantic space which allows computations based on conceptual meanings of terms rather than their surface forms.

0.21	0.08
0.04	-0.37
0.11	-0.39
0.37	0.15
0.63	-0.01
0.49	0.27
0.09	-0.51
0.25	0.11
0.28	-0.23
0.04	-0.26
0.11	-0.47

Table 8: Truncated Matrix V

	apple	blood	cells	ibm	data	desktop	tissue	graphics	memory	organ	plasma
C1	0.73	0.00	0.11	1.25	2.00	1.72	0.01	0.86	0.77	0.00	0.09
M1	0.00	1.18	1.27	0.00	0.33	0.00	1.63	0.00	0.85	0.84	1.51
C2	0.76	0.00	0.01	1.32	2.04	1.83	0.00	0.91	0.72	0.00	0.00
M2	0.00	1.08	1.19	0.00	0.49	0.00	1.52	0.00	0.86	0.77	1.41
M3	0.21	1.70	1.97	0.35	1.73	0.18	2.45	0.18	1.74	1.24	2.32
C3	0.73	0.15	0.39	1.25	2.17	1.68	0.35	0.85	0.98	0.17	0.41
C4	0.96	0.00	0.16	1.65	2.65	2.27	0.03	1.13	1.02	0.00	0.13
C5	1.22	0.00	0.00	2.11	3.21	2.95	0.00	1.46	1.08	0.00	0.00

Table 9: Document by Term Matrix after SVD

2.5 Clustering Algorithms

Clustering algorithms can be divided into three main groups, based on the methodology they employ. *Hierarchical methods* perform a series of merging or splitting operations to create clusters, while *partitional techniques* avoid pairwise operations and divide the set of objects into a given number of clusters, and then iteratively refine those clusters. *Hybrid methods* incorporate ideas from both.

Clustering algorithms can also be classified into three main categories based on the object representation they use. *Vector space* methods directly cluster feature vectors, while *similarity based* methods convert feature vectors into a similarity matrix where each cell contains a similarity measure for a pair of feature vectors. *Graph based* methods represent objects as graphs, and use graph partitioning techniques to cluster them. This thesis only concerns vector and similarity methods, but not graph based approaches. The interested reader is encouraged to consult [1], [42] for more information about the latter.

In all these methods, the number of clusters to be created can either be explicitly specified or automatically derived by the algorithm. We take the former approach in this thesis, although automatically determining the optimal number of clusters is an important area of future work.

In this thesis we limit our discussion to *hard clustering* algorithms that assign each object to at most one cluster. There are also soft (fuzzy) clustering methods [16] that determine the degree of membership of each object in each cluster, but those are not included here.

The following sections describe several widely used hierarchical and partitional methods, and shows that they can often be applied in either vector or similarity space.

2.5.1 Hierarchical

Hierarchical methods can be divided into two classes. *Agglomerative* methods merge a pair of clusters at each iteration, while *divisive* methods split a cluster into two at iteration.

Agglomerative methods start with each object in a separate cluster, so that if there are N objects, the algorithm begins with N initial clusters. The most similar clusters are merged during each iteration until the desired number of clusters are obtained. Divisive methods work in the opposite fashion and initially start with all objects in a single cluster. During each iteration, a cluster containing the least similar objects is split

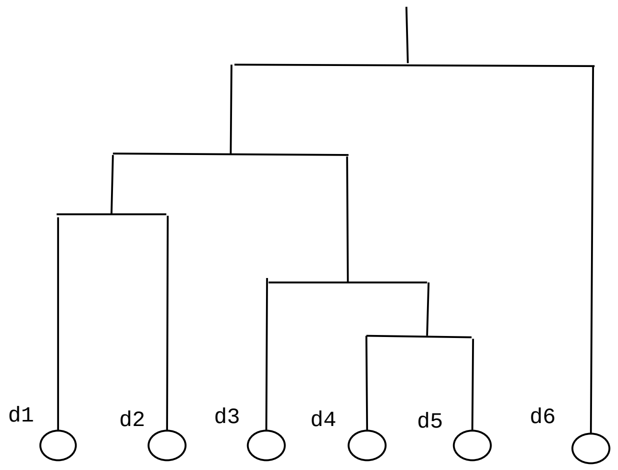

Figure 5: Example of Dendogram

into two. This continues until the required number of clusters are formed.

The clusterings found by agglomerative and divisive methods can be represented in a tree structure known as a *dendogram* that shows the clusters as found at each iteration of the algorithm. At the top-most level, the dendogram tree shows a single cluster containing all objects, while at the bottom-most level, there are as many leaf nodes as there are objects to be clustered. The dendogram tree can be used to retrospectively examine the progress of the clustering algorithm, and may shed insights into where clustering could be stopped to achieve optimal results.

Figure 5 shows an example of a dendogram tree where there are a total of 6 objects being clustered: {d1, d2, d3, d4, d5, d6}. Objects d4 and d5 are merged in the first iteration (read the tree in Figure 5 in bottom-up fashion). In the 2nd iteration of clustering, object d3 is merged with the cluster containing (d4, d5). Then, objects d1 and d2 are clustered in the next iteration to form a single cluster (d1, d2). After that, clusters (d1, d2) and (d3, d4, d5) are merged. Finally, object d6 joins the cluster (d1, d2, d3, d4, d5).

The decision as to which clusters should be split or merged during an iteration is dictated by a criteria function [44], which determines which clusters are most or least similar. The most widely used criteria functions for hierarchical methods are single link, complete link and average link.

In the agglomerative approach, the single link criteria chooses the pair of clusters with the minimum distance between their nearest members for merging, while, the complete link criteria selects the pair with the

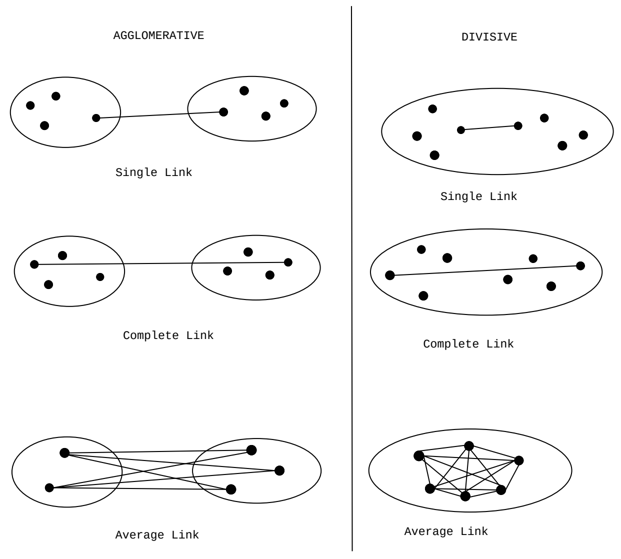

Figure 6: Single, Complete and Average Link Clustering

minimum distance between their farthest members. The average link method merges the pair of clusters that has the minimum average pairwise distance between the members. Note that objects with the minimum distance are considered to be the most similar.

In the divisive approach, the single link criteria chooses the cluster with the maximum distance between its nearest members for splitting while the complete link method selects the cluster with the maximum distance between its farthest members. The average link method splits the cluster with the maximum average pairwise distance between all pairs of its members.

Figure 6 shows how the single, complete and average link criteria select the pair of clusters for merging and a cluster for splitting. In the single link diagram, note that the two objects that are closest to each other are used for determining the amount of similarity between the two clusters, while in complete link it is the furthest pair of objects. Average link clustering measures the distance between the centroids of the two clusters to determine similarity.

Each of the two possible object representations can be employed in hierarchical clustering.

In vector space, there are N initial vectors for the N objects, each representing its own cluster. During each iteration, the clusters to be merged or split are selected according to the chosen criteria function. Note that the distance between a pair of vectors is determined by the angle between them, while similarity is measured by the cosine of this angle.

In similarity space, the input to the clustering algorithm is a similarity matrix that represents the pair–wise similarities between the given objects. Thus, N objects are represented in a N x N dimensional similarity matrix, whose rows and columns represent the initial N clusters. During each iteration, the clusters to be merged or split are chosen according to the selected criteria function. After merging, the similarity matrix is updated to show the similarities between the newly merged cluster and all other clusters in the collection according to the selected criteria function.

2.5.2 Partitional

Partitional algorithms divide the entire set of objects into a pre-determined number of clusters (say K) without going through a series of pair-wise merging or division steps. Unlike hierarchical methods, the clusters created during subsequent iterations are not related to those in the previous or next iterations. These methods are preferred on larger datasets due to their lower computational requirements ([9], [20]), as they do not require an exhaustive series of pairwise comparisons like the agglomerative methods do. The best known example of a partitional algorithm is the K-means clustering algorithm.

Partitional methods can be carried out on objects that are represented in vector or similarity space.

In vector space, the centroid of any cluster is the average of all the vectors that belong to that cluster. K-means initially selects K random vectors to serve as the centroids of the initial K clusters. It then assigns every other vector to one of the K clusters whose centroid is closest to that vector. After all vectors are assigned, the cluster centroids are re-computed by averaging all the vectors assigned to the same cluster. This repeats until convergence, that is until no vector changes its cluster across iterations, or in other words, when the centroids stabilize.

In similarity space, each object can be seen as a point in space such that the distance between any two points is a function of their similarity. Initially, each point is in its own cluster and represents the center of that cluster. During the first iteration, K-means selects K random points as K centers of the initial K clusters and assigns every other point to one of these K clusters that is closest to that point. Once all points are assigned to the clusters, the cluster centers are re-computed by taking the average of all points in the cluster. This is repeated until convergence.

2.5.3 Hybrid Methods

It is generally believed that the quality of clustering by partitional algorithms is inferior to that of the agglomerative methods. However, recent studies by [46], [45] have shown that these conclusions were based on limited experiments conducted with smaller data sets and that with larger data sets, partitional algorithms are not only faster but also lead to better results.

In particular Zhao and Karypis recommend a hybrid approach known as Repeated Bisections. This overcomes the main weakness of partitional methods, which is the instability in clustering solutions due to the choice of the initial random centroids. Repeated bisections method starts with a single cluster of all objects. At each iteration, a cluster whose bisection optimizes the chosen criteria function is selected for bisection. The cluster is bisected using the standard K-means method with K=2, while the criteria function maximizes the similarity between each object and the centroid of the cluster to which it is assigned [45]. As such this is a hybrid method that combines hierarchical divisive approach with partitional K-means method.

2.6 Evaluation

There are three techniques for evaluating the performance of clustering methods; external, internal and relative evaluation [16].

In this thesis, we focus on external evaluation metrics that determine clustering accuracy by comparing the solution against gold standard data for which true classification of the objects is available. This method of evaluation allows one to estimate in advance how the selected methodology will perform when used on real-life data whose true classification is unknown. While external evaluation requires knowledge of the true classification of objects, this information is not used at any point before evaluation, it is held out from the clustering or feature selection process.

Internal evaluation techniques are purely based on the metrics like intra-cluster and inter-cluster similarity and standard deviation among the clusters that do not use any knowledge about the existing categories. Relative evaluation techniques compare the solutions created by two different clustering methods. In this

	science	arts	sports	finance	R-Total
C1	2	0	3	10	15
C2	1	1	7	1	10
C3	6	1	1	2	10
C4	2	15	1	2	20
C-Total	11	15	12	15	55

Table 10: Confusion Matrix before Column Re-ordering

thesis we focus on external evaluation, and will not further discuss internal or relative evaluation techniques.

2.6.1 Column Reordering Method

Our external evaluation technique was suggested by [30]. It requires that we build a cluster by class distribution matrix which is known as a *confusion matrix.* The rows of this matrix represent the discovered clusters while columns represent the actual classes in the given gold standard. A cell value at (i,j) indicates the number of objects in the cluster represented by the i^{th} row that belong to the class represented by the j^{th} column.

Figure 10 is an example of a cluster by sense matrix which shows the distribution of 55 objects (i.e., the value in the last row and last column) in 4 clusters (C1, C2, C3, C4). The first row indicates that there are a total of 4 categories (science, arts, sports, finance) in gold standard data. The last column shows the row marginal totals, which are the total number of objects in each cluster. The last row shows the column marginals, which are the total number of objects belonging to each category in gold standard. Each cell value at (i,j) indicates the number of objects in the i^{th} cluster that belong to the category represented by the j^{th} column according to gold standard.

Accuracy is then computed by re-ordering the columns of the confusion matrix so that the diagonal sum is maximized. With each re-ordering, we indeed assign a class label to each cluster and then choose the mapping that results into the maximum number of objects in their true classes. The diagonal sum for each possible mapping scheme shows the number of objects in their correct classes if the clusters are labeled according to that mapping scheme. Table 11 shows the same cluster by sense distribution matrix shown in

	finance	sports	science	arts	R-Total
C1	10	3	2	0	15
C2	1	7	1	1	10
C3	2	1	6	1	10
C4	2	1	2	15	20
C-Total	15	12	11	17	55

Table 11: Confusion Matrix after Column Re-ordering

table 10 after reordering columns to get the maximal cluster–class mapping. Specifically, the matrix in table 11 shows that the maximal labeling scheme labels cluster C1 with *finance*, cluster C2 with *sports*, C3 with *science* and C4 with *arts*. In other words, cluster C1 represents the *finance* category, cluster C2 represents the *sports* category and so on...

The problem of finding the maximally accurate mapping of class labels to clusters is equivalent to the Assignment Problem in Operations Research or the Maximal Bipartite Matching problem in Graph Theory [18] [27].

2.6.2 Measuring Precision and Recall

Once the confusion matrix is created, the accuracy of clustering can be computed using precision and recall.

Precision is defined as the ratio of the number of objects in their correct classes divided by the number of objects attempted by the clustering algorithm. Remember that, an algorithm might not cluster all the given objects, especially the objects that do not share any feature with any other object will be put into singleton clusters. These are counted as the un-clustered objects during evaluation. Thus, the number of objects attempted by the algorithm (denominator in precision) is obtained by subtracting the un-clustered objects from the total number of objects. The number of objects correctly clustered (numerator) is the sum of the diagonal values of the column-reordered confusion matrix corresponding to the maximal mapping arrangement.

Thus, if we refer back to table 11, we can see that the diagonal sum is 38 (10 + 7 + 6 + 15), which indicates that 38 objects are in their correct categories. The total number of objects in the matrix is 55, which is the number of objects the algorithm placed in clusters. Given these values, precision is computed as follows:

$$precision = \frac{38}{55} = 0.69 \tag{16}$$

Recall is computed by dividing the number of objects correctly clustered by the total number of objects. Thus, recall is always less than or equal to the precision value, as the denominator of recall will always be greater than or equal to that of the precision, having the same numerator.

For the confusion matrix in 11, let's assume that there were a total of 57 objects and our clustering algorithm clustered only 55 of them. This means that there were 2 objects that our clustering algorithm was unable to place in a cluster. The total number of objects correctly clustered remains 38, and the total number of objects that were given to the algorithm is 57. Recall is then computed as follows:

$$recall = \frac{38}{57} = 0.67 \tag{17}$$

Precision and recall provide two separate measures of clustering performance. The F-measure is a single value that combines these two by taking two times the harmonic mean of precision and recall. Given the previous values of precision and recall, the F-measure is computed as follows:

$$\begin{aligned} F = 2 \times \frac{precision \times recall}{precision + recall} \\ F = 2 \times \frac{0.69 \times 0.67}{0.69 + 0.67} \\ F = 2 \times 0.34 \\ F = 0.68 \end{aligned}$$

2.6.3 Entropy and Purity

Entropy and purity are two fairly standard external evaluation measures that are used in clustering. We employ both of these after carrying out the column re–ordering method described above.

Entropy takes into account the distribution of objects in each cluster belonging to each class in gold standard.

A cluster made up predominantly of objects from a single class will have low entropy, while a cluster made up of a mixture of objects from multiple classes will have higher entropy.

The following formula computes the entropy of a cluster C_r made up of a total of n_r objects. n_r^i is the number of objects in cluster C_r that belong to the i^{th} class, while, *t* is the total number of classes in the gold standard.

$$E(C_r) = -\frac{1}{\log t}\sum_{i=1}^{t}\frac{n_r^i}{n_r}\log\frac{n_r^i}{n_r} \tag{18}$$

Assuming that there are a total of n objects grouped into k clusters then the following must be true:

$$\sum_{r=1}^{k} n_r = n \tag{19}$$

Thus, for the confusion table in 11, the value of n is 55, and t is 4. Then the entropy of each cluster can be computed as follows:

$$\begin{aligned} E(C_1) &= -\frac{1}{\log 4}[(\frac{10}{15}\times\log\frac{10}{15}) + (\frac{3}{15}\times\log\frac{3}{15}) + (\frac{2}{15}\times\log\frac{2}{15})] && (20)\\ &= -1.661\times(-0.1174 - 0.1398 - 0.1167) && (21)\\ &= 0.6211 && (22)\end{aligned}$$

$$\begin{aligned} E(C_2) &= -\frac{1}{4}[(\frac{3}{10}\times\log\frac{1}{10}) + (\frac{7}{10}\times\log\frac{7}{10})] && (23)\\ &= -1.661\times(-0.3 - 0.1084) && (24)\\ &= 0.6784 && (25)\end{aligned}$$

$$\begin{aligned} E(C_3) &= -\frac{1}{4}[(\frac{2}{10}\times\log\frac{2}{10}) + (\frac{2}{10}\times\log\frac{1}{10}) + (\frac{6}{10}\times\log\frac{6}{10})] && (26)\\ &= -1.661\times(-0.1398 - 0.2 - 0.1331) && (27)\\ &= 0.7855 && (28)\end{aligned}$$

$$E(C_4) = \frac{1}{4}[(\frac{4}{20} \times \log \frac{2}{20}) + (\frac{1}{20} \times \log \frac{1}{20}) + (\frac{15}{20} \times \log \frac{15}{20})] \tag{29}$$
$$= -1.661 \times (-0.2 - 0.0651 - 0.0937) \tag{30}$$
$$= 0.596 \tag{31}$$

The total entropy of the clustering solution is computed by taking the sum of the entropies of the individual clusters weighted by their sizes:

$$E_{total} = \sum_{r=1}^{k} \frac{n_r}{n} E(C_r) \tag{32}$$

From equations 22, 25, 28 and 31, we get the total entropy of clustering solution in table 11 as follows:

$$E = \frac{[(15 \times 0.6211) + (10 \times 0.6784) + (10 \times 0.7855) + (20 \times 0.596)]}{55}$$
$$= \frac{9.3165 + 6.784 + 7.855 + 11.92}{55}$$
$$= 0.6523$$

A perfect solution where each object is placed into its correct class will cause n_r^i = n_r for each cluster, leading to $E(C_r)$ = 0 for all r, and hence $E\_total = 0$.

Purity tells the degree to which a cluster represents the class used by the maximum number of its member objects.

$$P(C_r) = \frac{1}{n_r} max(n_r^i) \tag{33}$$

We obtain the following purity values for the confusion matrix in 11:

$$P(C_1) = \frac{10}{15} = 0.6667$$
$$P(C_2) = \frac{7}{10} = 0.7$$

$$P(C_3) = \frac{6}{10} = 0.6$$
$$P(C_4) = \frac{15}{20} = 0.75$$

The purity of entire solution is then computed as as the sums of the individual cluster's purity.

$$P_{total} = \sum_{r=1}^{k} \frac{n_r}{n} P(C_r) \tag{34}$$

From equation 34, we find the purity of the solution in table 11 to be:

$$P_{total} = \frac{10 + 7 + 6 + 15}{55} = 0.691 \tag{35}$$

The best solution, where n_r^i = n_r for each cluster, will lead to $P(C_r)$ = 1 for all r, hence, will have $P\_total =$ 1.

Note that purity is not equivalent to precision as described above, since precision requires that each class label is assigned to a unique cluster, while purity allows for the same class label assigned to multiple clusters. In fact, purity can be seen as a simple form of precision, where each cluster is simply assumed to represent the class to which most of its member objects belong.

3 Methodology

In this thesis, we use an unsupervised clustering approach to solve the problem of word sense discrimination. Instances of a target ambiguous word are observed in a raw corpus of text. Our goal is then to identify which instances refer to the same meaning of that word. As hypothesized by [25], instances referring to the same meaning will often use similar contextual words. Thus, word sense discrimination becomes the task of grouping together the instances of the target word that are used in similar contexts. The objects that we cluster are the contexts around the instances of a target word and the features that we use to represent these contexts are the words or word sequences frequently observed in the context of that word. Here, we assume that the scope of the contexts is limited to 1-2 sentences around an instance of the target word.

The following sections discuss the specific methodology of our solution.

3.1 Identifying Features from Raw Text

The instances of a target word whose contexts are to be clustered make up the *test data*. This is sometimes also referred to as the *evaluation data*. We want to represent the test instances by their most salient or discriminating features. There are a number of decisions that must be made in doing this. First, from what data will we identify features. Second, how will we actually extract the features from that data.

3.1.1 Feature Selection from Test or Training

The features used to represent test instances may be selected from that same test data, or those features may be obtained from a separate held–out corpus that is referred to as the *training data*. The decision as to whether to use the same data or held–out data for feature extraction depends on a number of factors.

If the test data is fairly large, and if there is no separate training data readily available, then it may make sense to simply identify the features from the test data. This is fairly common in clustering applications in general.

In our experiments, we use a separate training data, partially because we assume that the size of the test data will not be always sufficient to select a good set of features. Also, the use of held–out training data allows us to try different variations in the sources of training corpus, and see their effect on the performance of discrimination on the same test set.

For example, suppose we apply our discrimination methodology to the problem of email–classification, where we seek to group emails according to their topic. If we employed held–out training data, we could cluster new emails (say for year 2004) by selecting features from old emails (say from year 2003) in the hope to achieve similar organization. Or, if we didn't have enough emails to provide sufficient training, we could use some other larger email corpora such as the Enron Email Dataset (http://www-2.cs.cmu.edu/ enron/) or newsgroup corpora such as the 20-NewsGroups (http://people.csail.mit.edu/u/j/jrennie/public_html/20Newsgroups/) or the archives available from Google Groups (http://groups.google.com/). On the other hand, if we want to cluster all of our emails and we have a sufficiently large archive, then we could select features from the same data that is to be clustered. Both options are reasonable, and depend on the particular goals of the application.

3.1.2 Local versus Global Training

We call training data *target-specific* or *local* if every context in the training data includes the target word. Note that local training data corresponds to what is known as *lexical sample data* in the word sense disambiguation literature, where the corpus provides a sample of usages for a particular word. In general, if a large number of contexts that include a target word are available, then there are fairly clear advantages to using that for training data (as we will show in our experimental results).

The other variation that we tried in our experiments was to use a large amount of running text, such as newspaper corpora (e.g., the Associated Press Worldstream or New York Times) that consists of many complete articles and not just contexts around a specific target word. We call this method of using running text for feature selection as *global training*.

In local training each context consists of a sentence or perhaps paragraph that contains a single instance of the target word, while in global training a context might be a paragraph or entire article without respect to the particular words that occur therein. Thus, in global training it is nearly certain that there will be many contexts that do not include a particular target word.

As such a global training data provides general information about word usages, including those outside the contexts of a specific target word. We will show that this can provide a reasonably good level of performance, despite the fact that the training data is not specific to a particular word.

When using global training, all the data preprocessing and even some of the feature selection steps may be performed only once and then these preprocessed results can be used with various target words or for various kinds of experiments. Also, global training corpora are often easier to obtain than finding a large number of instances of a specific target word as is required by local training.

However, a possible disadvantage of global training is that it could introduce a large amount of noise in the feature set. This is because the general behavior and usage of words outside the contexts of a specific target word may not be always useful to discriminate among the senses of a particular word. However, the hope is that the broad coverage of global training will provide additional information not present in local training that will offset this disadvantage.

One of our objectives in this thesis is to study the effect of the nature of the training data on discrimination accuracy. For example, using the same test data for training, splitting all of the available data into training and test partitions, using some pre-classified training examples for clustering the new set of test instances, or using target-specific as well as global training data.

3.2 Types of Features

The features that we use to represent the context of the target word in the test data are all surface level lexical features. These are word–based features that can be observed directly in whatever text is serving as the source of features (be it the same test data or held–out training data). Specifically, we represent the context in which a target word occurs using unigrams, bigrams and co-occurrences.

3.2.1 Unigrams

Unigrams are single words that occur in the same context as the target word. Our use of unigrams is motivated by the success of *bag–of–words feature sets*, which are made up of all the words found in a sample of training data. Despite its simplicity, this feature set has proven successful in text classification and word sense disambiguation [26].

Obviously there are many words in text that do not provide information about meaning of the target word, in particular, function words like conjunctions, articles, and prepositions. As such we exclude these from our unigram feature set by specifying these words (which are to be ignored) in a *stop list*. Thus, in general we hope that unigram features will primarily be content words (noun, verbs, adjectives, and adverbs) that capture the meaning of a text.

In local training, where every training instance includes the target word, unigrams are the content words that occur above a given level of frequency in the context of the target word. If the training data is global, where every instance does not include a target word, unigrams are the most frequently occurring content words in that corpus.

In our local training experiments we apply a frequency cutoff of 2 by removing unigrams that appear only once in the training data.

Table 12 shows the top 20 most frequent unigrams (and their frequency of occurrence) as found in the training data for the verb *serve*, which is a part of the SENSEVAL-2 corpus. (Note that in this thesis we will use the diamond symbol <> to mark the end of a word.) We observe that some of the unigrams pertain to leadership positions and food, both of which relate to *serve* in some way. This shows how unigram features can in fact be useful for differentiating among meanings.

Table 13 shows the top 20 most frequent unigrams collected from all the SENSEVAL-2 training data (which includes *serve* plus 71 other words). Here we treat the SENSEVAL-2 training corpus as a source of global training and select unigram features without regard to any particular target word. As such the list of unigrams is fairly generic, and does not seem to suggest any particular meanings or topics. Hence, we do not employ unigram features when dealing with global training data.

3.2.2 Bigrams

Bigrams are typically defined as a consecutive sequence of two words. In this thesis, we extend that definition in two ways.

First, we allow a given window or number of words between the two words that make up the bigram. Thus, our bigram features are pairs of words that occur in a given order within some distance from each other in

Table 12: Local Unigrams for verb Serve

WORD<>	freq
president<>	194
hot<>	165
chairman<>	154
time<>	150
company<>	147
sauce<>	128
add<>	127
chief<>	122
cream<>	119
executive<>	107
million<>	102
old<>	102
minutes<>	99
chopped<>	97
purpose<>	96
butter<>	95
director<>	88
board<>	86
officer<>	84

Table 13: Unigram Features, Global Training

WORD<>	freq
time<>	1419
people<>	1022
way<>	931
day<>	892
work<>	824
old<>	745
still<>	712
head<>	700
long<>	687
life<>	622
man<>	594
house<>	592
world<>	557
right<>	554
million<>	552
market<>	548
home<>	519
come<>	518
same<>	515
high<>	509

Table 14: Bigrams, Window Size of 2

WORD1<>WORD2<>	n11	n1p	np1
TWINKLE<>TWINKLE<>	1	2	1
TWINKLE<>LITTLE<>	1	2	1
LITTLE<>STAR<>	1	1	1

the training corpus. We specify a window of size five, meaning that there could be at most three intervening words between the first and the second word that make up a bigram.

Second, we use the stop list to remove any bigrams that include at least one stop word. This is referred to an OR stop list, and is intended to restrict bigrams to being made up of two content words.

Suppose we identify and count all the bigram features in the following well known rhyme, where stop words are shown in lower case:

TWINKLE TWINKLE LITTLE STAR
how i WONDER what you are
up above the WORLD so HIGH
like a DIAMOND in the SKY

If we use a window size of two, and thereby consider bigrams to be two consecutive words, the resulting bigrams from this sample of text are shown in Table 14.

The numbers following the bigrams in Table 14 are (in order): the joint frequency n11 of the bigram, the marginal frequency n1p of WORD1, and the marginal frequency np1 of WORD2. The joint frequency tells how many times the bigram occurs in the corpus, while the marginal n1p tells how many bigrams there are that begin with WORD1. The marginal np1 indicates how many bigrams end with WORD2. In other words, n1p is the sum of all the n11 values of all bigrams whose first word is WORD1, while np1 is the sum of all the n11 values of all bigrams whose second word is WORD2.

Note that this is the complete list of bigram features for the sample above, since other observed two word

Table 15: Bigrams, Window Size of 5

WORD1<>WORD2<>	n11	n1p	np1
TWINKLE<>LITTLE<>	2	5	2
TWINKLE<>STAR<>	2	5	3
WORLD<>HIGH<>	1	1	1
STAR<>WONDER<>	1	1	2
DIAMOND<>SKY<>	1	1	1
LITTLE<>WONDER<>	1	2	2
LITTLE<>STAR<>	1	2	3
HIGH<>DIAMOND<>	1	1	1
TWINKLE<>TWINKLE<>	1	5	1

sequences (such as *how i* or *i WONDER*) are not considered since they include one or two stop words.

Expanding the window size has a fairly dramatic effect. For example, if we use a window of size 5 (and thereby allow up to 3 intervening words between WORD1 and WORD2), we obtain the set of bigrams shown in Table 15.

Note that we do not allow bigrams that span across the boundaries of a context. This means that bigrams in which the two constituent words belong to different contexts are not counted. Thus far in our example we have been considering the entire Twinkle, Twinkle rhyme to be one context. However, if we treat each line as a single context and do not allow bigrams to cross line/context boundaries, we will get a different set of bigrams as shown in Table 16.

Notice that the bigrams *LITTLE<>WONDER, STAR<>WONDER* and *HIGH<>DIAMOND* do not appear here as their component words appear on separate lines, which are now treated as separate contexts.

It should be noted that while we don't require bigrams to be consecutive words, we do require them to retain their original ordering since this often has a strong impact on meaning. For example, *sharp<>razor* and *razor<>sharp* are distinct bigrams and may refer to different underlying meanings. Bigram features allow us to retain those kinds of distinctions. Our use of bigrams is motivated by their recent success as features

Table 16: Bigrams, Window Size of 5, Each Line a Context

WORD1<>WORD2<>	n11	n1p	np1
TWINKLE<>LITTLE<>	2	5	2
TWINKLE<>STAR<>	2	5	3
WORLD<>HIGH<>	1	1	1
TWINKLE<>TWINKLE<>	1	5	1
DIAMOND<>SKY<>	1	1	1
LITTLE<>STAR<>	1	1	3

Table 17: Contingency Table for Bigrams

	WORD2	-WORD2	RSUM
WORD1	n11	n12	n1p
-WORD1	n21	n22	n2p
CSUM	np1	np2	npp

in word sense disambiguation [29].

Measures of Association for Bigrams Each bigram and its associated counts of the form

WORD1<>WORD2<>n11 n1p np1

can be converted into a 2 by 2 contingency table as shown in Table 17. Here, n11, n12, n21 and n22 are referred to as the observed frequencies while n1p, n2p, np1, np2 are the marginal frequencies.

What follow are the specific descriptions of what each cell in this table represents.

- n11 = number of times a bigram WORD1<>WORD2 is observed
- n12 = number of bigrams in which WORD1 is at the first position but WORD2 is not at the second position

- n21 = number of bigrams in which WORD2 is at the second position but WORD1 is not at the first position
- n22 = number of bigrams in which WORD1 is not at the first position and WORD2 is not at the second position
- n1p = total number of bigrams in which WORD1 is at the first position (n11+n12)
- np1 = total number of bigrams in which WORD2 is at the second position (n11+n21)
- n2p = total number of bigrams in which WORD1 is not at the first position (n21+n22)
- np2 = total number of bigrams in which WORD2 is not at the second position(n22+n12)
- npp = total number of bigrams in the sample = sum of n11 scores of all bigrams

Given the marginal frequencies n1p, np1, n2p, np2, we can estimate the expected values for these values based on the assumption that the two words are occurring in the corpus independently as follows:

- m11 = expected value of n11 = (n1p*np1/npp)
- m12 = expected value of n12 = (n1p*np2/npp)
- m21 = expected value of n21 = (n2p*np1/npp)
- m22 = expected value of n22 = (n2p*np2/npp)

Having computed all the marginal, observed and expected frequencies, we then compute the log–likelihood ratio for each bigram as follows:

$$G^2 = 2\sum_{ij} n_{ij} \times \log \frac{n_{ij}}{m_{ij}} \tag{36}$$

This measures the deviation between the observed and expected frequencies, and if a pair of words shows a high deviation between these values, the words are shown not to be independent. A formal test of significance can be performed by selecting a p-value (we normally use 0.05) and seeing if the resulting score is

Table 18: Contingency Table for Bigram TWINKLE<>STAR

	STAR	-STAR	RSUM
TWINKLE	n11=2	n12=n1p-n11=3	n1p=5
-TWINKLE	n21=np1-n11=1	n22=np2-n12=2	n2p=npp-n1p=3
CSUM	np1=3	np2=npp-np1=5	npp=8

greater than the corresponding critical value (3.841). This critical value comes from the chi–square distribution, based on 1 degree of freedom. Then, any bigrams whose log–likelihood score is greater than this value will be considered as a feature (and those below will be discarded). Those bigrams with scores above the critical value are considered to be strongly associated, and are not occurring together due to some chance occurrence.

From the example in Table 16, we show the computation of the log–likelihood ratio for the pair

TWINKLE<>STAR<>2 5 3

Note that the value of npp is 8, which can be determined by summing all of the n11 values. With this information, we can construct a complete 2 x 2 contingency table for bigram TWINKLE<>STAR as shown in Table 18.

Given these observed values, the expected values can be estimated as follows:

$$m11 = (n1p * np1/npp) = 3 * 5/8 = 1.875$$
$$m12 = (n1p * np2/npp) = 5 * 5/8 = 3.125$$
$$m21 = (n2p * np1/npp) = 3 * 3/8 = 1.125$$
$$m22 = (n2p * np2/npp) = 5 * 3/8 = 1.875$$

Then the log–likelihood ration for the bigram TWINKLE<>STAR can be computed as follows:

$$\begin{aligned}G^2 &= 2\sum_{ij} n_{ij} \times \log \frac{n_{ij}}{m_{ij}}\\
&= 2 \times [(n11 * \log \frac{n11}{m11}) + (n12 * \log \frac{n12}{m12}) + (n21 * \log \frac{n21}{m21}) + (n22 * \log \frac{n22}{m22})]\\
&= 2 \times [(2 * \log \frac{2}{1.875}) + (3 * \log \frac{3}{3.125}) + (1 * \log \frac{1}{1.125}) + (2 * \log \frac{2}{1.875})]\\
&= 0.0358\end{aligned}$$

Note that this value falls well below our critical value of 3.841, so we might discard that as a feature. However, note that this example is artificially small, usually our value of npp is much larger (in the thousands or perhaps even millions).

Similarly, we compute the log–likelihood ratio for all other bigrams found in Table 16 and show those results in Table 19.

Table 19: Bigram Log–Likelihood Scores

WORD1<>WORD2<>	score
WORLD<>HIGH<>	6.0283
DIAMOND<>SKY<>	6.0283
TWINKLE<>LITTLE<>	2.2672
LITTLE<>STAR<>	2.2092
TWINKLE<>TWINKLE<>	1.0243
TWINKLE<>STAR<>	0.0358

Notice that on arranging the bigrams in the descending order of their log–likelihood ratio, we get a different ordering than if we ordered them according to their frequency count, n11. For instance, bigrams like *WORLD<>HIGH* or *DIAMOND<>SKY* get higher log–likelihood ratios despite their smaller counts. On the other hand, bigram *TWINKLE<>STAR* gets the least score in spite of its higher n11 count. This is because the log–likelihood ratio doesn't only consider the joint frequency n11, but it compares all the observed frequencies with their corresponding expected values.

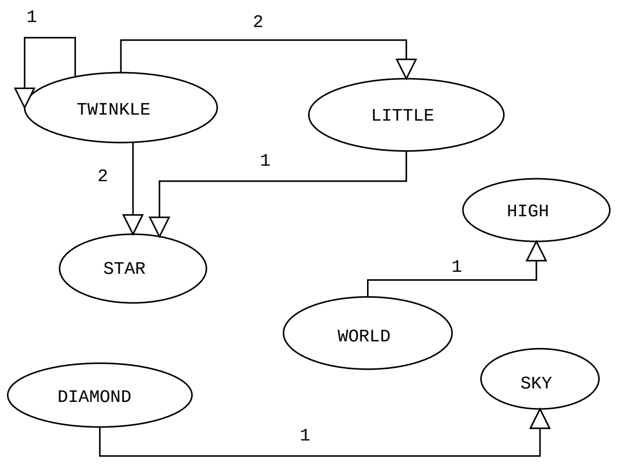

Figure 7: Graphical Representation of Bigrams

Graphical Models of Bigrams Figure 7 shows a graphical representation of bigrams that we refer to as a bigram graph. Each vertex of this graph represents a word, and an edge joining vertex X to Y represents the bigram X<>Y. The edges are weighted and the weights may indicate either the frequency counts or statistical scores of association between the corresponding pair of words. In this figure, we use the simple joint frequency counts of bigrams as weights. Notice that edges X<>Y and Y<>X could have different scores as they represent different bigrams.

For each bigram of the form *WORD1<>WORD2<>n11 n1p np1* shown in Table 16, the marginal frequency of WORD1 (n1p) is the same as the out–degree of WORD1 in the corresponding bigram graph shown in Figure 7. Recall that this is the sum of n11 counts of all bigrams having WORD1 at the first position. Similarly, the marginal frequency of WORD2 (np1) is the same as the in–degree of WORD2 in the bigram graph. This indicates the sum of n11 counts of all bigrams that have WORD2 at the second position.

Vector Space Model of Bigrams The bigrams as shown in figure 7 are internally stored in an adjacency matrix, which is formatted as shown in Table 20. We refer to this as a bigram matrix. The rows of a bigram matrix represent the words that have a non-zero out-degree in the corresponding bigram graph, while columns represent the words that have a non–zero in–degree. The cell value at (i,j) in the bigram graph shows the score associated with the bigram WORDi<>WORDj, since the i^{th} row represents WORDi and the j^{th} column represents WORDj. Thus, each i^{th} row of the bigram matrix can be viewed as a bigram

Table 20: Bigram Matrix

	LITTLE	STAR	HIGH	TWINKLE	SKY
TWINKLE	2	2	0	1	0
WORLD	0	0	1	0	0
DIAMOND	0	0	0	0	1
LITTLE	0	1	0	0	0

Table 21: Bigram Statistics Matrix

	LITTLE	STAR	HIGH	TWINKLE	SKY
TWINKLE	2.2672	0.0358	0	1.0243	0
WORLD	0	0	6.0283	0	0
DIAMOND	0	0	0	0	6.0283
LITTLE	0	2.2092	0	0	0

vector of the WORDi whose j^{th} index shows the score of the bigram WORDi<>WORDj. Since we use frequency counts of bigrams in this example, the bigram vectors are represented in an integer–valued vector space whose dimensions are made up of the words represented across the columns.

Table 21 shows the matrix for the bigrams from Table 16 based on log–likelihood scores rather than frequency counts. A bigram matrix that uses such statistical scores is referred to as a bigram statistics matrix. Each row of a bigram statistics matrix can be viewed as a vector in a real-valued word space.

3.2.3 Co–Occurrences

Co–occurrences are similar to bigram features, except that they are not ordered. Two words are called co-occurrences of each other if they occur within some specified window of each other without regard to their order. In our experiments, we set this window size to 5, allowing at most three intervening words between the two words to call them as co-occurrences. While for bigrams we say that WORDi<>WORDj occurs n times and WORDj<>WORDi occurs m times, for co–occurrences we simply say that WORDi and WORDj

Table 22: Bigrams, Window Size of 3, each Line as Context

WORD1<>WORD2<>	n11	n1p	np1
SELLS<>SEA<>	1	3	1
SELLS<>SHELLS<>	1	3	2
SEA<>SHELLS<>	1	2	2
SEA<>SHORE<>	1	2	1
SHELLS<>SELLS<>	1	1	1
SELLS<>SALES<>	1	3	1
SALES<>SMELL<>	1	1	1

co-occur (n+m) times. Like bigrams, we do not allow co–occurrences that contain one or two stop words. Consider the following example, where stop words are indicated in lower case:

she SELLS SEA SHELLS on the SEA SHORE
SHELLS she SELLS on SALES SMELL

Suppose, we first find the possible bigrams in this text using a window of size 3, and where each line represents a distinct context (meaning that bigrams may not cross line/context boundaries). The set of bigrams that results is shown in Table 22.

Now, suppose, we are simply interested in finding which words co-occur regardless of their order. Using the same window of 3 and ignoring pairs that span across the lines/contexts, we find the set of co-occurrence pairs as shown in Table 23.

Each word pair WORD1<>WORD2 as shown in the co-occurrence list (figure 23) is followed by three values n11, n1p and np1, which have similar but slightly different interpretations here. n11 shows the total number of times the two words co-occur together irrespective of the ordering. Notice that, *SHELLS* follows *SELLS* once on line 1 in the given text, while, *SELLS* follows *SHELLS* once on line2. Hence, the co-occurrence file lists the n11 score of the pair *SHELLS<>SELLS* as 2 unlike the bigram file that shows two separate orderings as *SHELLS<>SELLS* and *SELLS<>SHELLS* each with score of 1. The n1p count

Table 23: Co–occurrences, Window Size of 3, each Line as Context

WORD1<>WORD2<>	n11	n1p	np1
SELLS<>SEA<>	1	4	3
SELLS<>SHELLS<>	2	4	3
SEA<>SHELLS<>	1	3	3
SEA<>SHORE<>	1	3	1
SELLS<>SALES<>	1	4	2
SALES<>SMELL<>	1	2	1

of a co-occurrence pair WORD1<>WORD2 shows the sum of the n11 counts of all pairs in which WORD1 appears at either position. For example, the n1p count of word *SEA* is 3 as it appears in total 3 pairs each with the n11 score of 1. Similarly, the np1 count is the sum of all word pairs in which WORD2 appears.

Measures of Association for Co-occurrence Each co-occurrence pair of the form *WORD1<>WORD2<>n11 n1p np1* is converted into a 2 x 2 contingency table similar to the one shown in the previous section (Table 17). However, the observed and marginal frequencies have slightly different meanings for co-occurrences, as explained below.

- n11 = number of times WORD1 and WORD2 co-occur in either order
- n12 = number of word pairs in which WORD1 occurs but WORD2 doesn't.
- n21 = number of word pairs in which WORD2 occurs but WORD1 doesn't.
- n22 = number of word pairs in which neither WORD1 nor WORD2 occurs.
- n1p = total number of pairs in which WORD1 occurs. (n11+n12)
- np1 = total number of pairs in which WORD2 occurs. (n11+n21)
- n2p = total number of pairs in which WORD1 doesn't occur. (n21+n22)
- np2 = total number of pairs in which WORD2 doesn't occur. (n22+n12)

Table 24: Co–Occurrence Log–Likelihood Scores

WORD1<>WORD2<>	score
SALES<>SMELL<>	2.9690
SEA<>SHORE<>	1.9225
SELLS<>SEA<>	1.2429
SEA<>SHELLS<>	0.1965
SHELLS<>SELLS<>	0.1965
SELLS<>SALES<>	0.0580

- npp = total number of word pairs in the sample = sum of n11 scores of all co-occurrence pairs.

Here, we only consider the pairs of words that occur within a certain distance or window from each other. The expected frequencies have the same definitions as they do for bigrams, and the actual calculation of log–likelihood is carried out the same way. Table 24 shows the log–likelihood ratios that we result from the co–occurrences listed in Table 23.

Graphical Model of Co-occurrences Figure 8 shows the graphical representation of the co-occurrences in what we call a co-occurrence graph. Similar to the bigram graph, each vertex of this graph represents a word. However, the edges of the co-occurrence graph are undirected as the order of words that co-occur is not important. An edge connecting vertices X and Y simply indicates that words X and Y co-occur within the specified distance from each other. In this graph, we use the log–likelihood ratios between the word pairs as the weights on the edges, rather than their frequency counts.

Notice here that, the n1p or np1 frequency of any word is same as its degree in the undirected co-occurrence graph, which indicates the total number of pairs in which the word occurs regardless of its position.

In word sense discrimination, we are often interested in co-occurrences of a target word. As can be noticed in the co-occurrence graph (figure 8), co-occurrences of a particular word are all the words connected to it in the co–occurrence graph. For example, *SHELLS, SALES, SEA* are co–occurrences of *SELLS*.

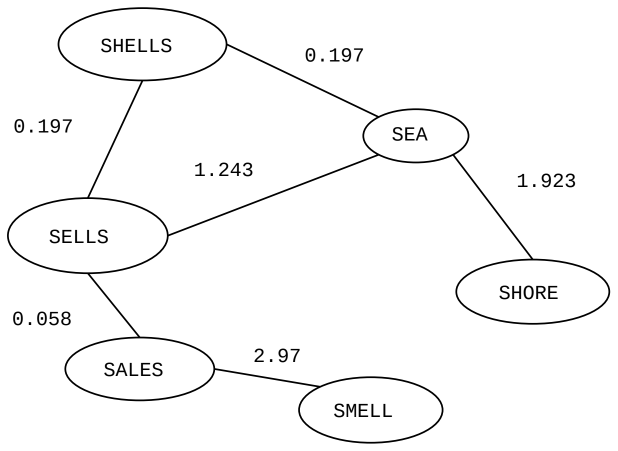

Figure 8: Graphical Representation of Co-occurrences

Table 25: Co-occurrence Matrix

	SELLS	SEA	SHELLS	SHORE	SALES	SMELL
SELLS	0	1.243	0.197	0.058	0	0
SEA	1.243	0	0.197	1.923	0	0
SHELLS	0.197	0.197	0	0	0	0
SHORE	0	1.923	0	0	0	0
SALES	0.058	0	0	0	0	2.97
SMELL	0	0	0	0	2.97	0

Vector Space Model of Co-occurrences We store co-occurrences in a matrix called a co-occurrence matrix, whose rows and columns represent the words and cell entries indicate the co-occurrence scores of the corresponding word pairs.

Each word encountered in the text is assigned a unique index and represents the row and column of the co-occurrence matrix at that index. The matrix entries then show the co-occurrence score between the corresponding pair of words. Table 25 shows the same co-occurrence pairs as in figure 8 in the matrix format. As the same word represents the row/column at any index and as the value at (i,j) is same as that at (j,i), the co-occurrence matrix is always square and symmetric.

Kth Order Co-Occurrences Co-occurrences as defined thus far can be seen as the words that are directly connected to each other in a word co–occurrence graph. In other words, the words that are co–occurrences of each other are joined by a path of length 1, or are at a distance of one edge away from each other. In this section, we extend that view of co-occurrences and define K^{th} Order Co-occurrences as words that are K edges away from each other in the co-occurrence graph.

[39] introduced the idea of second order co-occurrences as words that co-occur with the co-occurrences of a target word. We observed that these are the words that are indirectly connected to the target word via one of its co-occurrences. In other words, second order co-occurrences are two edges (with one intermediate node) away from the target word.

In Figure 8, *SHELLS* and *SHORE* are second order co-occurrences of each other since they are connected via exactly one word: *SEA*. Assuming that *SELLS* is a target word here, *SHORE* and *SMELL* will then become the second order co-occurrences of the word *SELLS*.

Similarly, we define the K^{th} order co-occurrences of a target word as those words that are connected to the target word by exactly K edges (or K-1 intermediate nodes). In general, we call any two words connected by K edges as the K^{th} order co-occurrences of each other. For example, in the co-occurrence graph shown in Figure 8, words *SHORE* and *SALES* are called the third order co-occurrences, as the shortest path connecting them has length 3 (with two intermediate nodes). Similarly, *SHORE* and *SMELL* are forth order co-occurrences as they are four edges away from each other.

Given this framework, co-occurrences as defined in the beginning of this section can be seen as the first-order co-occurrences as those are connected by exactly one edge.

3.3 Context Representations

After selecting features from the training data, we turn our attention to the test data that contains the instances of the target word to be clustered. Each instance in the test data is converted into a form that indicates which of the features occur (or not). In particular, the context around each instance of the target word in the test data is represented as a vector of features called a context vector. We refer to the operation of identifying which features occur in the test instance as *feature matching*, since we take a list of features found in training data, and check if they occur in the test instances by performing an exact string match.

In this section, we discuss two types of context representations: first order context vectors as used by [30] and second order context vectors as introduced by [39].

3.3.1 First-Order Context Vectors

A first order context vector indicates which of the features directly occur in the context of the test instance. If the feature values are binary, a context vector simply indicates which features occur in the context. If the features are non-binary, a context vector indicates the number of times each feature is matched in the context.

The text below was created by randomly selecting web search results for the query ”sells NEAR shells” using the Alta-Vista search engine [15]. The NEAR directive requests any pages that contain ‘*sells* and *shells* within 10 positions of each other be returned. As such this data contains occurrences of these words in both possible orders, which allows us to effectively illustrate the nature of co–occurrences.

> *SHERRY SELLS CANDLES OF SEA SHELLS BY THE SEA SHORE*
> *SHE ALSO SELLS BRIDAL JEWELRY OF SHELLS*
> *SHELLS SHE SELLS ARE SEA SHELLS, I AM SURE*
> *IF SHE SELLS SHELLS ON THE SEA SHORE, WHICH SHE DOES*
> *THEN THE SHELLS OUGHT TO BE THE SEA SHORE SHELLS*
> *WE SPECIALIZE IN SEA SHELL CANDLES, CHRISTMAS ORNAMENTS AND DECORATIONS*
> *SHE SELLS SEA SHELLS EARRINGS AT SHERRY’S SHELL STORE*

Unigram Matching We perform exact feature matching rather than fuzzy matching, and hence it is required that a feature word be matched literally in the context of a test instance. Also note that we do not attempt to stem (reducing words to their base forms) or normalize either the training or test data, so a feature word does not even match with its own morphological variants in the context.

In the Alta-Vista sample of text, the unigram *SHELL* will be matched total 2 times on lines 6 and 7, as shown by bold entries below.

SHERRY SELLS CANDLES OF SEA SHELLS BY THE SEA SHORE

SHE ALSO SELLS BRIDAL JEWELRY OF SHELLS

SHELLS SHE SELLS ARE SEA SHELLS, I AM SURE

IF SHE SELLS SHELLS ON THE SEA SHORE, WHICH SHE DOES

THEN THE SHELLS OUGHT TO BE THE SEA SHORE SHELLS

WE SPECIALIZE IN SEA **SHELL** *CANDLES, CHRISTMAS ORNAMENTS AND DECORA-TIONS*

SHE SELLS SEA SHELLS EARRINGS AT SHERRY'S **SHELL** *STORE*

However, the unigram *SHELLS* is matched a total of 8 times as shown below.

SHERRY SELLS CANDLES OF SEA **SHELLS** *BY THE SEA SHORE*

SHE ALSO SELLS BRIDAL JEWELRY OF **SHELLS**

SHELLS *SHE SELLS ARE SEA* **SHELLS**, *I AM SURE*

IF SHE SELLS **SHELLS** *ON THE SEA SHORE, WHICH SHE DOES*

THEN THE **SHELLS** *OUGHT TO BE THE SEA SHORE* **SHELLS**

WE SPECIALIZE IN SEA SHELL CANDLES, CHRISTMAS ORNAMENTS AND DECORA-TIONS

SHE SELLS SEA **SHELLS** *EARRINGS AT SHERRY'S STORE*

Thus, the unigram *SHELL* does not match the context that has its plural form *SHELLS* and vice-a-versa. Similarly, the verb feature *SELL* will not match its different forms like *SELLS* or *SOLD*.

Our hypothesis is that dimensionality reduction techniques like SVD when performed on a large sample of text will tend to smooth irregularities and variations in language usage, including the different morphological forms of words.

Bigram Matching A bigram of the form *WORD1<>WORD2* selected using a window of size n (say n=5) is said to be matched in the text only if WORD1 is followed by at most n-2 words (here 3 words if n=5) in the same context before WORD2 appears.

Suppose we do not use any window (count only consecutive word pairs) while selecting a bigram feature *SEA<>SHELLS*. Then, the above sample text will have 3 matches of bigram *SEA<>SHELLS* where *SEA* is immediately followed by *SHELLS* as shown below.

SHERRY SELLS CANDLES OF **SEA SHELLS** *BY THE SEA SHORE*
SHE ALSO SELLS BRIDAL JEWELRY OF SHELLS
SHELLS SHE SELLS ARE **SEA SHELLS**, *I AM SURE*
IF SHE SELLS SHELLS ON THE SEA SHORE, WHICH SHE DOES
THEN THE SHELLS OUGHT TO BE THE SEA SHORE SHELLS
WE SPECIALIZE IN SEA SHELL CANDLES, CHRISTMAS ORNAMENTS AND DECORATIONS
SHE SELLS **SEA SHELLS** *EARRINGS AT SHERRY'S SHELL STORE*

Note that our exact matching does not match the sequence *SEA SHORE SHELLS* on line 5, as we require the words *SEA* and *SHELLS* to occur consecutively when windowing is not used. We also do not consider the sequence *SEA SHELL* on line 6 as it is not an exact match of the feature *SEA SHELLS*. The sequence *SHELLS ON THE SEA* on line 4 is discarded for same reason; *SEA* and *SHELLS* in this sequence do not occur in the same order as expected by the feature *SEA SHELLS* nor do they appear consecutively.

Suppose that we allow a window of 5 i.e. at most 3 intervening words, then the bigram *SELLS<>SHELLS* will match the sample text 5 times as shown below.

SHERRY **SELLS** *CANDLES OF SEA* **SHELLS** *BY THE SEA SHORE*
SHE ALSO **SELLS** *BRIDAL JEWELRY OF* **SHELLS**
SHELLS SHE **SELLS** *ARE SEA* **SHELLS**, *I AM SURE*
IF SHE **SELLS SHELLS** *ON THE SEA SHORE, WHICH SHE DOES*
THEN THE SHELLS OUGHT TO BE THE SEA SHORE SHELLS
WE SPECIALIZE IN SEA SHELL CANDLES, CHRISTMAS ORNAMENTS AND DECORATIONS
SHE **SELLS** *SEA* **SHELLS** *EARRINGS AT SHERRY'S SHELL STORE*

Co-occurrence Matching A co-occurrence feature selected using a window size of n (here n=5) is said to be matched in a context if there are at most n-2 (here 3) words between that feature word and the target word in the context. While matching, it doesn't matter whether a feature word follows or precedes the target word. Thus, a co-occurrence feature that appears within a given window from the target word in the training data is said to be matched in the context of a test instance if it appears within the same distance from the target word in that context.

For example, in the context of the word *SHELLS*, word *SEA* appears a total of 6 times within a window of size 5, as shown by the bold face entries below.

SHERRY SELLS CANDLES OF **SEA** *SHELLS BY THE* **SEA** *SHORE*
SHE ALSO SELLS BRIDAL JEWELRY OF SHELLS
SHELLS SHE SELLS ARE **SEA** *SHELLS, I AM SURE*
IF SHE SELLS SHELLS ON THE **SEA** *SHORE, WHICH SHE DOES*
THEN THE SHELLS OUGHT TO BE THE **SEA** *SHORE SHELLS*
WE SPECIALIZE IN SEA SHELL CANDLES, CHRISTMAS ORNAMENTS AND DECORA-TIONS
SHE SELLS **SEA** *SHELLS EARRINGS AT SHERRY'S SHELL STORE*

Note that on line 3 there are two instances of *SHELLS*, and *SEA* is within 5 positions of both these instances. However, we require that there is only single instance of the target word in each context. Even if there are multiple occurrences of the same word in a context, we treat only one of them as the target word. This is because we do not make any assumption that multiple instances of a word in the same context always refer to the same meaning. Its possible to find the same word used with different meanings in the same context. For example, *SHELLS SHE SELLS ARE SEA SHELLS AND NOT FIREWORK SHELLS.* In order to identify a correct instance of the target word, we mark it with an XML tag like $< head > SHELLS < /head >$.

On the similar lines, on line 5, *SEA* will match only if the second instance of *SHELLS* is marked as the head/target word, as there are four words between *SEA* and the first instance of *SHELLS* which goes beyond our window size of 5 that allows for only 3 intervening words.

Table 26: Unigram Features, First Order Example

WORD<>
SEA<>
GUNS<>
SHORE<>
SYSTEM<>
CORALS<>
EXECUTE<>
EXPLODE<>
COMMANDS<>
FILE<>
STORE<>
FIREWORK<>
UNIX<>

Building Actual First Order Contexts Suppose we selected the unigram features shown in Table 26 from some hypothetical training data.

Further suppose that our test data consists of following text, where the beginning and end of each context is marked with XML $< context >$ tags, and the target word is marked with the $< head >$ tag. For simplifying further explanation, each context appearing on line *i* is given an identifier C_i as shown in the beginning of each line :

C1: $< context >$ *a* $< head > SHELL < /head >$ *SCRIPT is a FILE of UNIX COMMANDS* $< /context >$
C2: $< context >$ *if she SELLS SHELLS BY the SHORE, then i am SURE she SELLS SEA SHORE SHELLS and not the FIREWORK* $< head > SHELLS < /head >< /context >$
C3: $< context >$ *STORE the CSH COMMANDS in a* $< head > SHELL < /head >$ *and INVOKE CSH to EXECUTE these COMMANDS* $< /context >$
C4: $< context >$ *FIREWORK* $< head > SHELLS < /head >$ *EXPLODE onto the USU-*

Table 27: First Order Context Vectors

	SEA	GUNS	SHORE	SYSTEM	CORALS	EXECUTE	EXPLODE	COMMANDS	FILE	STORE	FIREWORK	UNIX
C1	0	0	0	0	0	0	0	1	1	0	0	1
C2	1	0	2	0	0	0	0	0	0	0	1	0
C3	0	0	0	0	0	1	0	2	0	1	0	0
C4	0	0	0	0	0	0	1	0	0	0	1	0
C5	0	1	0	0	0	0	0	0	0	0	0	0
C6	0	0	0	0	1	0	0	0	0	0	0	0
C7	1	0	1	0	0	0	0	0	0	1	0	0

ALLY DARK SCREENS in a VARIETY of COLORED STREAKS $< /context >$

C5: $< context >$ *ARTILLERY GUNS were USED to FIRE HIGHLY EXPLOSIVE* $< head >$ $SHELLS < /head >< /context >$

C6: $< context >$ *we OFFER the BEST COLLECTION of* $< head > SHELLS < /head >$ *and CORALS at VERY REASONABLE PRICES* $< /context >$

C7: $< context >$ *the LARGEST SEA* $< head > SHELL < /head >$ *STORE on the SHORE* $< /context >$

For each of the above contexts, we construct a first order context vector that indicates which features occur in that context. Table 27 shows the first order context vectors for each of the above contexts. Specifically, each line of Table 27 shows a vector representing the context that appears on the corresponding line in the above text, where the selected features become the dimensions of these vectors. A cell value at (i,j) indicates the number of times a feature represented by the j^{th} column is found matched in the context represented by the i^{th} row.

Table 28: Unigram Features, Second Order Example

WORD<>
SEA<>
GUNS<>
SYSTEM<>
COMMANDS<>
STORE<>
UNIX<>

3.3.2 Second-Order Context Vectors

Second order context vectors indirectly represent the context of a test instance by an average of the word vectors of features that match in the context. First, a bigram or co–occurrence matrix is constructed to show the frequency counts or statistical scores of association between all pairs of words that form bigrams or co-occurrences in the training data. Thus, each word in the training data is represented by a bigram or co-occurrence vector that shows how often (or how strongly) it is associated with each of the other words in the training corpus. These vectors are referred to as *feature vectors* or just *word vectors*. The context of each test instance is then represented by the average of all feature vectors of words that make up that context (where low frequency and stop words are excluded).

Suppose Table 28 shows the features we selected from the training data.

And suppose Table 29 shows feature vectors for each of the words included in this feature set. Specifically, each row shows a co-occurrence vector for the corresponding feature word shown in column 1. The values in the matrix are the log–likelihood ratio between the corresponding pairs of words. For this example, these values are obtained from the Associated Press Worldstream corpus. The words representing the dimensions (columns) in this table are a randomly selected subset of the unigram features found in this corpus. We do not show all the dimensions for this corpus as it consists of more than 100,000 columns!

Having created the feature vectors for each of the selected feature word in the training data, we turn our attention to building a second order context vector for each instance in the test data.

Table 29: Example Feature Word Vectors

	SELLS	WATER	MACHINE	COMPUTERS	DOS	BOMBS
SEA	18.5533	3324.9846	0	0	0	8.7399
GUNS	22.4537	287.4839	85880.9155	48.1596	0	699.5669
SYSTEM	0	6858.3415	1.7013	14.5221	116.1319	0
COMMANDS	0	0	0	31.1924	90.2884	0
STORE	134.5102	205.5469	0	62.3774	0	0
UNIX	0	0	116.6942	109.1895	0	0

Table 30: Feature Vectors of UNIX, COMMANDS and SYSTEM

	SELLS	WATER	MACHINE	COMPUTERS	DOS	BOMBS
SYSTEM	0	6858.3415	1.7013	14.5221	116.1319	0
COMMANDS	0	0	0	31.1924	90.2884	0
UNIX	0	0	116.6942	109.1895	0	0

Suppose we want to create a second order context vector for the instance below:

> $< context >$ *the C* $< head >$ $SHELL$ $< head >$ *OFFERS a UNIX USER many DIFFERENT COMMANDS. on any UNIX SYSTEM, ONLINE DOCUMENTATION of these COMMANDS is AVAILABLE in MAN PAGES.* $< context >$

As before, stop words are written in lower case letters, the target word is marked in $< head >$ tags, and the $< context >$ tags indicate the beginning and end of the context.

Note that in this context, we observe features UNIX and COMMANDS twice and feature SYSTEM once. Other features like SEA, GUNS, STORE do not appear. We obtain the feature vectors for the words UNIX, COMMANDS and SYSTEM from the co-occurrence matrix in table 29.

Then we can compute the second order context vector as follows, where $\vec{WORD}$ stands for the vector associated with a given word, and the numerator represents the sum of all the vectors associated with the words in the context:

$$\frac{(2* \overrightarrow{UNIX}) + (2* \overrightarrow{COMMANDS}) + (\overrightarrow{SYSTEM})}{5}$$

$$= \frac{2*(0 \quad 6858.3415 \quad 1.7013 \quad 14.5221 \quad 116.13190) + 2*(0 \quad 0 \quad 0 \quad 31.1924 \quad 90.2884 \quad 0) + (0 \quad 0 \quad 116.6942 \quad 109.1895 \quad 0 \quad 0)}{5}$$

$$= \frac{(0 \quad 13716.7 \quad 3.4026 \quad 29.0442 \quad 232.264 \quad 0) + (0 \quad 0 \quad 0 \quad 62.3848 \quad 180.577 \quad 0) + (0 \quad 0 \quad 116.6942 \quad 109.1895 \quad 0 \quad 0)}{5}$$

$$= \frac{(0 \quad 13716.7 \quad 120.097 \quad 200.618 \quad 412.841 \quad 0)}{5}$$

$$= (0 \quad 2743.34 \quad 24.0194 \quad 40.1236 \quad 82.5682 \quad 0)$$

Note that a second order context vector indirectly represents contexts in terms of the words that co-occur with the contextual words rather than with the target word. The intuition behind this indirection is that a co-occurrence or bigram vector of a word is assumed to capture the meaning of that word in terms of the words that most frequently co-occur with it, and by averaging the word vectors of contextual words, we try to capture the meaning of the entire context in terms of the meanings of its contextual words. In short, we try to represent the meaning of a context as an average meaning of the words that appear in the context. As our method is knowledge–lean, we try to find the meanings of feature words found in any context by identifying the words that most commonly co-occur with them in the training data.

3.4 Singular Value Decomposition

Matrices as shown in Tables 20, 25 or 27 can be reduced in dimensions by performing Singular Value Decomposition (SVD). Specifically, we use the single vector Lanczos method (las2) as implemented in SVDPACKC [2]. We selected this particular algorithm based on the results reported by [2] that show its overall performance efficiency over other methods.

The program las2 expects the input matrix to be in the Harwell-Boeing format (see [2]) along with the values of various program options specified separately in a parameter file *lap2*. Various constants for the program are specified at compilation time via the header file las2.h. In the discussion that follows we describe the various parameters in files lap2 and las2.h, and how we set their values in our experiments.

3.4.1 Setting parameters in lap2

The parameter file lap2 allows the user to specify the values of various parameters while running las2. The format of the lap2 file is as follows:

name lanmax maxprs endl endr vectors kappa

where all parameter values are listed on a single line and are separated by blank space.

The parameters of most relevance to us are the 2nd and the 3rd, i.e., lanmax and maxprs. Other than these two (lanmax and maxprs), all others are set to their default values as given in the original lap2 file distributed with SVDPACKC. $< name >$ specifies the name of the matrix and we set it to a descriptive string that indicates the data source and specific settings used to create the matrix.

$< maxprs >$ specifies the number of singular triplets to be discovered by las2. In our experiments, we set maxprs to *min(K, COLS/RF)* where K = 300 and RF = 10. K and RF are both referred to as the reduction factors. The value of K specifies the number of significant dimensions to be retained by SVD while RF specifies the scaling factor by which the original dimensions are reduced. For example, if the matrix prior to SVD has 1200 columns, then we reduce it down to min(300, 1200/10) = min(300,120) = 120 dimensions. On the other hand, if the matrix has 12,000 columns, we retain only the top 300 ones (as min(300, 12000/10) = min(300,1200) = 300).

Apart from $< maxprs >$, we also set the $< lanmax >$ parameter that specifies the number of iterations for las2. As suggested by the SVDPACKC authors, we set this to (3 * maxprs) which is said to be enough to compute $< maxprs >$ singular values. The value of $< lanmax >$ has to be at least as high as the $< maxprs >$ parameter and the values of $< lanmax >$ and $< maxprs >$ can not exceed the total number of columns in the original matrix. Hence, we set $< lanmax >$ to min(3*maxprs, #cols) to make sure that it is always less than or equal to the number of columns (#cols) of the given matrix.

3.4.2 Setting constants in las2.h

The header file las2.h sets the values of various program constants to las2.

- NMAX - Specifies the maximum possible number of columns in the matrix given to las2. By default, las2.h will have NMAX = 3000 that allows maximum 3000 columns in the matrix. As this default is too small for most of our experiments, we set NMAX to 10,000 that allows up to 10,000 features. In case of our experiments with large (global) training data, we even set this constant to something as high as 200,000. In short, this value is expected to be higher than the number of columns in the matrix prior to SVD.

- NZMAX - Specifies the maximum possible number of non-zero values in the matrix. Default settings in las2.h have NZMAX = 100000. Assuming that our local 10,000 x 10,000 matrix is approximately 1% dense, we set NZMAX to 1,000,000 (10,000 x 10,000 / 100). Our experiments with large data used NZMAX of about 5,000,000.

- LMTNW - This specifies the maximum total memory to be allocated by las2. Default value of LMTNW in las2.h is 600000. Specifically, LMTNW is expected to be at least as high as (6*NMAX + 4*NMAX + 1 + NMAX*NMAX). Again by Assuming that our 10,000 x 10,000 matrix is 1% dense, we set LMTNW to 10,010,000. The global experiments with a very large training used LMTNW = 2,000,100,000.

3.5 Determining Number of Clusters to Create

A significant challenge in any clustering task is to determine how many clusters should be created for the given data. While discriminating senses of a word, we face a similar question: how many senses does a word actually have? As we use a knowledge–lean approach here, we can not simply refer to an electronic dictionary to find out this answer. The following discussion describes the various strategies we used to overcome this challenge in our experiments.

3.5.1 Similarity Score Cutoff

The parameter, number of clusters to be created, specifies a terminating condition to a clustering algorithm that will continue to cluster until the required number of clusters are formed. Recall that an agglomerative algorithm starts with N initial clusters (for N test instances) and merges the most similar pair of clusters at each iteration. A divisive algorithm starts with a single cluster and splits a cluster with most dissimilar elements in each iteration.

Thus, if no stopping conditions are specified, an agglomerative algorithm will continue merging until a single cluster is formed, while a divisive algorithm will continue splitting until the given N instances are divided into N clusters. In our first experiments (described as Experiment 1 in the Experimental Results section), we used a strategy that sets the number of clusters to be discovered to 1 for agglomerative and to N (where N = number of test instances) for a divisive method. Then we used another stopping rule to terminate clustering before these conditions are met.

Specifically, we set threshold values on similarity scores such that an agglomerative algorithm stops if no pair of clusters has similarity above the specified cutoff and a divisive algorithm stops if no two members of any cluster have similarity below the specified cutoff. Agglomerative methods stop due to a lack of enough similarity between any pair of clusters for merging, while a divisive method stops due to lack of enough dissimilarity within any cluster for splitting. However, we realized that determining the appropriate score at which to stop clustering is a challenging problem in its own right and there is no single score value that finds suitable number of clusters for any dataset.

3.5.2 Filtering Low Frequency Senses

After realizing the difficulty in setting a suitable score cutoff for clustering, we decided to pre–process the data such that it includes instances of only the top N most frequent senses of any word. Thus, irrespective of the number of senses a word has in a dictionary, we selected only the N most frequent senses. This allowed us to create exactly the same number of clusters as the word senses. We refer to this method of filtering data based on the rank of the word senses as a *rank based sense-filter*. In brief, we count the number of instances for each word sense in the evaluation data (i.e., test data where the true classification of an instance is known), then rank the word senses in the descending order of these frequencies and finally select the top N most frequent senses by removing the instances that use the senses ranking below N.

The limitation of the rank based sense-filter is that a selected value of N could be too high for some of the words that have fewer popular senses. This poses another challenge of discriminating very rare senses of such words. Here, we assume that the given test data is a representative sample of the general language usage and the percentage frequency of any word sense remains more or less same in any other representative sample. Thus, popular senses are supposed to be the ones that are used by a large number of instances while the rare senses are those used by very few instances in the given evaluation sample. Clustering the instances of rare word senses is assumed to be more challenging because there will be fewer instances of such rare senses in any both training and test data. This results into fewer features characterizing such senses, which in turn leads to very sparse context representations of the instances that use these rare senses.

Since the rank based sense-filter poses the challenge of including some very rare word senses, in our later experiments we selected the most frequent senses based on some percentage frequency cutoffs instead of rankings. In other words, we selected only those senses that are illustrated by at least M% of the total instances in the evaluation data. M is typically set to some number greater than 5 or 10.

For example, assuming that the evaluation data has at least 100 total instances, by setting M to 10, we make sure that there are at least 10 instances of each sense. This type of preprocessing based on the percentage frequencies of word senses is referred to as the *percent based sense-filter*. Thus, from a given test data with total N instances, we remove the instances which are tagged with the senses that in turn are used by less than NM/100 instances. Hence, we know that every word sense in the filtered test data has at least NM/100 instances. We then create some arbitrarily large number of clusters that is greater than expected number of senses for any word.

For example, assuming that most words have approximately 3-6 senses, we create 10 clusters. Our hypothesis here is that, a good discrimination experiment will automatically create approximately same number of clusters as the true senses and the extra clusters will contain very few instances. In the ideal case, we expect that each extra cluster will have a single instance. Thus, we can ignore the singleton clusters (containing only one instance) without loosing many instances. The other variation that we tried was to ignore the clusters that contain less than NM/100 instances. This is because we know that our M-percent filter has only retained the senses with at least NM/100 instances.

As such we expect each of the discovered clusters to have at least NM/100 instances. In practice, however, the distribution of instances in the discovered clusters is never exactly same as that of the true senses in the data. Hence, after applying a M% filter, we usually ignore the clusters containing less than some P instances, where P is slightly less than NM/100.

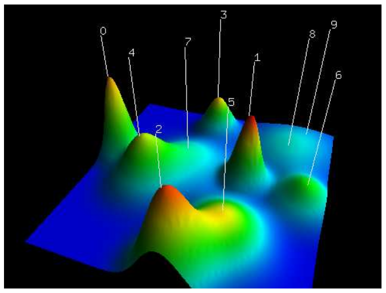

Figure 9: Graphical Visualization of Clusters

3.5.3 Cluster Visualizations

Both the approaches described above require knowledge of the true sense tags of the test instances prior to clustering and hence can't be applied to the problem of clustering when this knowledge is not available. In other words, the kind of preprocessing proposed above based on the frequency of word senses is only useful for experimental purposes when we know the true distribution of senses in the given data. Hence, we believe that the strategy we employ for determining the correct number of clusters should not use any knowledge beyond what is available in raw text.

There are tools available such as GCLUTO that produce a graphical visualization of discovered clusters that helps to determine which clusters are significant. More specifically, the 3D mountain view of clusters as created by GCLUTO shows each cluster by a mountain in a 3D plane. The distance between any two mountains is a inverse function of their inter-cluster similarity while the height of a mountain is directly proportional to the intra-cluster similarity. Thus, one can easily notice that the short clusters are the extra ones and in fact can be viewed as the parts of one of the major (taller) clusters that is closest to them.

Figure 9 illustrates the case when the gold-standard evaluation data has fewer senses than the actual number of clusters discovered. In this case, we requested 10 clusters but the mountain view reveals that there are only 5 to 7 significant clusters and hence the data should be better divided into around 6 clusters.

The problem with this method is that it requires manual inspections of cluster results and is not fully automatic. Hence, the problem of automatically discovering the correct number of clusters without using any knowledge outside the given raw corpus remains a challenge that we plan to address in our future work.

Table 31: Cluster by Sense Confusion Matrix

	SERVE10	SERVE12	SERVE2	SERVE6
C0:	401	23	42	9
C1:	16	5	9	4
C2:	3	1	8	4
C3:	21	8	23	9
C4:	147	66	75	15
C5:	8	0	2	1
C6:	0	2	1	2
C7:	0	3	8	3
C8:	9	16	12	7
C9:	67	393	161	124

3.6 Evaluation

The performance of a clustering algorithm can be evaluated using gold–standard data in which instances being clustered are manually attached with their true sense tags. Some of the instances can have multiple sense tags attached by same or different human taggers, which means that they use more than one sense of the target word. However, we always used hard clustering in our experiments and hence do not classify any instance into more than one clusters. Since this poses a challenge in evaluating such instances with multiple answers, we remove all but the most frequently used sense attached to those instances that have multiple possible correct answers. In other words, every instance in our evaluation data is tagged with only one sense which is the most frequent of all the senses attached to it.

The output of a clustering algorithm shows clusters of given instances such that all the instances that use the same meaning of the target word are grouped together in the same cluster. If we know the true sense tags of the instances belonging to various clusters, we can construct a cluster by sense distribution matrix as shown in Table 31 that shows the number of instances of each sense in gold–standard in each of the discovered clusters.

The Table 31 suggests that we discovered total 10 clusters (represented by the rows) but there were only 4 senses (shown by the columns) in the gold–standard. Each cell value at (i,j) indicates the number of instances with true sense S_j as represented by the j^{th} column that are members of the cluster C_i as represented by the i^{th} row. In general, we will build a M x N confusion matrix if we create M clusters and there are total N senses in the gold–standard.

We then try to determine which sense of the target word is most closely represented by each cluster. A cluster containing maximum number of instances using a specific sense is assumed to represent that sense. Thus, the problem of determining what sense each cluster represents turns out to be a typical assignment problem of mapping sense tags to clusters. We assume one sense per cluster and hence do not make any attempt to attach multiple senses to a single cluster or a single sense to multiple clusters.

3.6.1 Eight Rooks (not Queens) Evaluation Algorithm

This problem of mapping senses to clusters is similar to the famous 8 Queens problem in chess. This attempts to place 8 queens on a standard 8 x 8 chess board such that no two queens kill each other. As queens can travel in any direction (horizontal, vertical or diagonal), the problem in fact tries to place queens such that no two queens are in the same row, column or diagonal in the 8 x 8 matrix.

We observed that if we attach multiple senses to a single cluster, we in fact select two cells in the same row in the confusion matrix. And similarly, we will select two cells from the same column if we attach a single sense to multiple clusters. Hence, what we would like to achieve here is to select cells from the confusion matrix as shown in Table 31 such that not two cells fall in either same row or column. Thus, we view our M x N confusion matrix as a M x N rectangular chess board on which we attempt to place min(M,N) rooks such that no two rooks kill each other. In chess, rooks travel only horizontally and vertically and therefore the condition that no rook should kill another rook automatically achieves our goal of not selecting any two cells from the same row or column.

Each possible solution to this problem in fact gives one possible assignment of senses to clusters. For example, selecting a confusion cell at (p, q) is equivalent to labeling cluster C_p with the sense S_q. The value of this cell at (p, q) is essentially the number of instances in cluster C_p that will be in their correct sense class S_q if cluster C_p is assumed to represent sense S_q. Thus, if we add the values at all the selected cells

Table 32: Confusion Matrix when (M = N)

	SERVE10	SERVE12	SERVE2	SERVE6
C0:	556	89	119	25
C1:	16	5	9	4
C2:	3	1	8	4
C3:	97	422	205	145

for a particular mapping, we get the total number of instances correctly classified if clusters are assumed to represent the senses as suggested by that mapping. For finding the accuracy of the solution, we select the maximal accurate mapping that gives the maximum number of total instances in their correct sense classes.

In most of our experiments, we do not assume the knowledge of actual number of senses for a word. Hence, the number of clusters we end up creating for a word could be more, less or equal to the actual number of senses that word has as per the gold–standard. The following is a discussion of how we handle these three cases while assigning senses to clusters for evaluation.

3.6.2 #CLUSTERS (M) = #SENSES (N)

Suppose we create exactly same number of clusters as the senses in the gold–standard as illustrated in Table 32. Thus, the N x N confusion matrix in this case will result into total N! (factorial) possible mappings that label every cluster with a single sense.

The following shows the most accurate mapping for the confusion matrix in Table 32:

C0=>SERVE10

C1=>SERVE2

C2=>SERVE6

C3=>SERVE12

This mapping suggests that we view Cluster C0 as sense SERVE10, cluster C1 as sense SERVE2 and so on.

Table 33: Confusion with (M = N) after Column Re-ordering

	SERVE10	SERVE2	SERVE6	SERVE12
C0:	556	119	25	89
C1:	16	9	4	5
C2:	3	8	4	1
C3:	97	205	145	422

Here, SERVE10, SERVE2, etc. are sense tags/identifiers used to denote each sense in the gold–standard. How these sense tags map to actual dictionary meanings depends on the particular source of gold–standard data which we do not discuss here.

We then re-arrange the columns of the confusion matrix such that the cells selected by the most accurate mapping fall across the diagonal as shown in Table 33. The diagonal entries of the confusion matrix after column reordering indicate the number of instances in each cluster that are in their correct sense class. Thus, the sum of all diagonal values gives the overall accuracy for this solution. In this case, it is (556+9+4+422)=991 which means a total of 991 instances are correctly classified according to the best mapping.

Table 34 shows the final report that we create for our evaluation and inspections of the results. The additional column of TOTAL shows the total number of instances in each cluster and the values in the brackets are the percentage of the total instances belonging to that cluster. This is referred to as the column of row marginal values as the value in each row of this column is in fact the sum of all other values on the same row.

Similarly, the additional row of TOTAL indicates the column marginals which is the total number of instances using the sense shown by the particular column along with their corresponding percentage distributions shown in the brackets.

The total number of instances actually clustered is simply the sum of all entries in the matrix which is 1708 in this case. Hence, the precision is 991/1708 = 58.08 for this particular experiment. The total number of instances that we actually gave to the clustering algorithm here was (1708+44)= 1752 as indicated by the denominator in recall. This means that our clustering algorithm wasn't able to cluster 44 instances. Hence, the recall turns out to be (991/1752)=56.56 which is slightly less than the precision.

Table 34: Final Report for Confusion case (M=N)

	SERVE10	SERVE2	SERVE6	SERVE12	TOTAL	
C0:	556	119	25	89	789	(46.19)
C1:	16	9	4	5	34	(1.99)
C2:	3	8	4	1	16	(0.94)
C3:	97	205	145	422	869	(50.88)
TOTAL	672	341	178	517	1708	
	(39.34)	(19.96)	(10.42)	(30.27)		

Precision = 58.02(991/1708)

Recall = 56.56(991/1708+44)

3.6.3 #CLUSTERS (M) > #SENSES (N)

When we create more clusters than the actual number of senses in the gold–standard (M > N), our mapping solutions that assume one sense per cluster, leave the (M-N) clusters unlabeled. In other words, N senses can be mapped only to N (out of M) clusters. We expect that a good methodology should automatically create the right number of clusters leaving very few (ideally only 1) instances in the extra (M-N) clusters. Each possible mapping selects a subset made up of N clusters from the total M clusters and then solves the mapping problem similar to the previous case when M = N.

Table 35 illustrates the case when we create more (10) clusters than the actual number of senses (4) in the gold–standard. Table 36 shows the same confusion table after re-ordering its columns to reflect the maximum accurate mapping across the diagonal.

This shows that only clusters C0, C3, C4 and C9 are assigned labels according to the best mapping scheme. The rest of the clusters are assumed to be the extras which are expected to contain a smaller percentage of the total instances. Hence, the total number of instances that are correctly classified is again the sum of the diagonal values of the column–reordered confusion matrix which is (401+9+75+393)=878. Note that, this value does not take into account the instances in the remaining (so called extra) clusters. In an ideal

Table 35: Confusion Matrix when (M > N)

	SERVE10	SERVE12	SERVE2	SERVE6
C0:	401	23	42	9
C1:	16	5	9	4
C2:	3	1	8	4
C3:	21	8	23	9
C4:	147	66	75	15
C5:	8	0	2	1
C6:	0	2	1	2
C7:	0	3	8	3
C8:	9	16	12	7
C9:	67	393	161	124

Table 36: Confusion with (M > N) after Column Re-ordering

	SERVE10	SERVE6	SERVE2	SERVE12
C0:	401	9	42	23
C3:	21	9	23	8
C4:	147	15	75	66
C9:	67	124	161	393

Table 37: Final Report for Confusion case (M > N)

	SERVE10	SERVE6	SERVE2	SERVE12	TOTAL	
C0:	401	9	42	23	475	(27.81)
C3:	21	9	23	8	61	(3.57)
C4:	147	15	75	66	303	(17.74)
C9:	67	124	161	393	745	(43.62)
C1:*	16	4	9	5	34	(1.99)
C2:*	3	4	8	1	16	(0.94)
C5:*	8	1	2	0	11	(0.64)
C6:*	0	2	1	2	5	(0.29)
C7:*	0	3	8	3	14	(0.82)
C8:*	9	7	12	16	44	(2.58)
TOTAL	672	178	341	517	1708	
	(39.34)	(10.42)	(19.96)	(30.27)		

Precision = 55.43(878/1584)

Recall = 50.11(878/1708+44)

solution, all extra clusters are expected to be singleton and hence we will loose only (M-N) instances from our accuracy score.

A report displaying the column–reordered confusion along with the marginal totals of all clusters is shown in table 37. Note that, the rows marked with * show the extra clusters that aren't labeled by the best mapping. The instances belonging to these extra clusters are treated as un–clustered and hence not considered in precision. Note that the extra clusters occupy a small percentage of the overall instances and hence ignoring them will not have much impact on the overall accuracy.

Table 38: Confusion Matrix when (M < N)

	SERVE10	SERVE12	SERVE2	SERVE6
C0:	556	89	119	25
C1:	116	428	222	153

Table 39: Confusion with (M < N) after Column Re-ordering

	SERVE10	SERVE12
C0:	556	89
C1:	116	428

3.6.4 #CLUSTERS (M) < #SENSES (N)

The confusion matrix in Table 38 demonstrates the third possibility when we create fewer clusters than actual number of senses in the gold–standard, i.e., M < N. In this case, our labeling strategy that assumes one sense per cluster, will attach only M (out of N) senses to M clusters, leaving the extra (N-M) senses free. In other words, we do not make any attempt to tag more than one sense to any cluster.

Table 39 shows the same confusion matrix from Table 38 after re-ordering its columns, to show the best mapping across the diagonal.

This shows that the best mapping used only two senses SERVE10 and SERVE12. The score of the best mapping is (556+428)=984. Though, we do not take into account the instances of extra senses (here, SERVE2 and SERVE6) in accuracy score, we do count them in both precision and recall. In other words, unlike we do in case when M > N, we do not discard the instances of extra senses as un–clustered.

Table 40 shows the final report created for the same solution. The columns marked with # indicate the extra senses that weren't attached to any cluster while determining the accuracy.

Table 40: Final Report for Confusion case (M $<$ N)

	SERVE10	SERVE12	SERVE2#	SERVE6#	TOTAL	
C0:	556	89	119	25	789	(46.19)
C1:	116	428	222	153	919	(53.81)
TOTAL	672	517	341	178	1708	
	(39.34)	(30.27)	(19.96)	(10.42)		

Precision = 57.61(984/1708)

Recall = 56.16(984/1708+44)

4 Experiments

We have thus far introduced various types of features, context representations, similarity measures and clustering algorithms. Now we can address one of the central questions of this thesis:

> Does any particular type of feature, context representation, similarity measure, or clustering algorithm give optimal results under all or certain circumstances?

By circumstances, we mean anything in the formulation of the data or algorithm that could have an effect on the overall accuracy of word sense discrimination. For example, this could include (but is not limited to) the following:

- The different types and amounts of data to be discriminated,
- the nature of the training training (local versus global),
- coarse versus fine granularity in word meanings, and
- balanced versus skewed distribution of word senses in the training/test data.

To that end, we present the results of four separate experiments, each of which is designed to compare and contrast the various choices that can be made when formulating a word sense discrimination solution.

4.1 Experiment 1: Lexical Features and Similarity Measures

The objective this experiment is to determine to what extent different types of first order features and similarity measures impact the accuracy of sense discrimination when performed in similarity space.[1]

These experiments use the first order representations of contexts with unigram, bigram and second order co-occurrences as features. In addition to using each of these types of features separately, we also created a novel feature type we call the *mix* by taking the union of the feature sets of these three features.

[1]These experiments were performed in December 2002, and were originally published in the Student Research Workshop at HLT-NAACL 2003 [34].

Clustering was then carried out in similarity space by computing pairwise similarities among the first order context vectors, using the matching and cosine similarity coefficients. We used the UPGMA clustering method, which is equivalent to average link clustering and McQuitty's Similarity Analysis. The experiments are carried out with local training data where each training example used for selecting features includes an instance of the target word.

4.1.1 Data

In Experiment 1, we used two well known sources of sense–tagged text, the LINE data [21] and the English lexical sample data from the SENSEVAL-2 comparative exercise among word sense disambiguation systems [11].

The LINE data contains 4,146 instances, where each consists of two to three sentences around a single occurrence of the word *line*. Each instance has been manually tagged with one of six possible senses. We randomly selected 100 instances of each sense for test data, and a separate set consisting of 200 instances of each sense for training. This gives a total of 600 evaluation instances, and 1200 training instances. This is done to test the quality of our discrimination method when senses are uniformly distributed and where no particular sense is dominant.

The standard distribution of the SENSEVAL-2 data consists of 8,611 training instances and 4,328 test instances. There are 73 distinct target words found in this data: 29 nouns, 29 verbs, and 15 adjectives. Most of these words have less than 100 test instances, and approximately twice that number of training examples. Instances in the training data indicate the correct sense of the word intended in that context, while, the correct senses for the test instances are provided in a separate answer key file. Each word has from 8 to 12 possible senses according to both the keyfile and the training data. Also, there are many training and test instances in this data that show multiple correct sense tags.

It's important to note that we do not use the sense tags available in the training and test data as a part of the clustering process, nor do we use them for feature selection. However, we do use them to filter out low frequency senses of words that appear in the training and test set. We believe this is a reasonable step to take, since the amount of training and test data is very small, and yet the number of senses per word is relatively large. This leads to some senses that occur a very small number of times, and we eliminate the smallest of these prior to any processing going forward.

Specifically, we only retain those training and test instances whose actual sense is among the top five most frequent senses as observed in the training data for that word. We believe that even 5 is an aggressive number of senses for a discrimination system to attempt, considering that Pedersen and Bruce [30] experimented with 2 and 3 senses, and Schütze [39] made binary distinctions.

Also, in cases where a test instance has more than one possible correct answer, we only kept the most frequent of those. The other possible answers were discarded, since our evaluation technique assumes one possible correct sense per cluster. We also noticed that the SENSEVAL-2 data identifies target words that are being used as proper nouns. We decided to not this fact in our discrimination by removing these so–called P tags from the data. After carrying out all these preprocessing steps, we were left with total 7,476 training and 3,733 test instances.

We then specify an upper limit on the number of senses that our clustering algorithm should discover. In Experiment 1, we set this limit to 5 for SENSEVAL-2 words and 6 for LINE. As we included all senses at every rank above 5, without trying to break the ties, the actual number of senses in the evaluation data could be 5 or more. Also, we set a similarity score cutoff to 0 which stops clustering as soon as there is no pair of clusters with similarity above 0. This cutoff sometimes stops clustering prematurely before creating 5 clusters. Hence, even on selecting top 5 senses and creating 5 clusters, the actual number of discovered clusters is not always same as the number of true senses in the evaluation data.

4.1.2 Results

For each word in the SENSEVAL-2 data and LINE, we conducted various permutations of Experiment 1, each of which uses a different combination of features and measure of similarity. Specifically, we performed total 8 experiments on each word using all combinations of 4 types of features (unigrams, bigrams, second-order co-occurrences and mix) with 2 types of similarity measures (matching and cosine coefficients).

As this leads to a very large number of results to analyze (73 words * 8 experiments/word = 584), we computed the average accuracy obtained for all words belonging to the same part of speech (POS) category (nouns, adjectives and verbs). We also counted the number of words from each POS for which a particular experiment did better than the majority classifier. As the word LINE uses larger amount of training and test data than SENSEVAL-2 words, we do not include LINE in the noun category of SENSEVAL-2 words and analyze its results separately.

Table 41 displays the average precision and recall for each POS category for SENSEVAL-2 words, and Table 42 shows the same for LINE.

The first column indicates the POS, the second shows the feature type, the third lists the measure of similarity, the fourth and the fifth show the average precision and recall of all words in that POS category for the particular experiment shown by the 2nd and 3rd columns. The sixth column shows the average percentage of the majority sense for all words in that POS. The final column shows the actual number of words in the given POS that gave accuracy greater than the percentage of the majority sense of that word for a particular combination of feature type and similarity measure as indicated by columns 2 and 3.

4.1.3 Analysis

Tables 43, 44 and 45 show the breakdown of SENSEVAL-2 results by part of speech (POS). Specifically, each value in these tables indicates the number of words from the particular POS on which a particular experiment (as indicated by the corresponding row and column labels) performed better than the majority classifier.

Recall that there were total 29 nouns, 28 verbs and 15 adjectives. As these tables indicate, the performance was overall better for verbs and nouns. But hardly any adjectives showed results better than the majority classifier. We believe that, this could be because the adjectives in this data have skewed distributions that results in a very high accuracy attained by the majority classifier, which makes this a difficult standard for an unsupervised method to reach. Verbs and nouns, on the other hand, have fairly balanced distributions which suggests that our strategy works better on the data in which no particular sense dominates.

Tables 46, 47 and 48 show a similar breakdown of results for SENSEVAL-2 words organized by the feature type. Each table value shows the number of words from the POS indicated by the corresponding row, on which a particular combination of feature type and similarity measure performed better than the majority classifier.

These results show that the second order co-occurrences (SOCs) and unigrams achieved the overall best

Table 41: Experiment 1 Results : Features and Similarity Measures (SENSEVAL-2)

pos	feat	meas	prec	rec	maj	$>$ maj
noun	soc	cos	0.49	0.48	0.57	6/29
		mat	0.54	0.52	0.57	7/29
	big	cos	0.53	0.50	0.57	5/29
		mat	0.52	0.49	0.57	3/29
	uni	cos	0.50	0.49	0.57	7/29
		mat	0.52	0.50	0.57	8/29
	mix	cos	0.50	0.48	0.57	6/29
		mat	0.54	0.51	0.57	5/29
verb	soc	cos	0.51	0.49	0.51	11/28
		mat	0.50	0.47	0.51	6/28
	big	cos	0.54	0.45	0.51	5/28
		mat	0.53	0.43	0.51	5/28
	uni	cos	0.42	0.41	0.51	13/28
		mat	0.43	0.41	0.51	9/28
	mix	cos	0.43	0.41	0.51	12/28
		mat	0.42	0.41	0.51	7/28
adj	soc	cos	0.59	0.54	0.64	1/15
		mat	0.59	0.55	0.64	1/15
	big	cos	0.56	0.51	0.64	0/15
		mat	0.55	0.50	0.64	0/15
	uni	cos	0.55	0.50	0.64	1/15
		mat	0.58	0.53	0.64	0/15
	mix	cos	0.50	0.44	0.64	0/15
		mat	0.59	0.54	0.64	2/15

Table 42: Experiment 1 Results : Features and Similarity Measures (LINE)

word	feat	meas	prec	rec	maj	$>$ maj
line	soc	cos	0.25	0.25	0.17	1/1
		mat	0.23	0.23	0.17	1/1
	big	cos	0.19	0.18	0.17	1/1
		mat	0.18	0.17	0.17	1/1
	uni	cos	0.21	0.21	0.17	1/1
		mat	0.20	0.20	0.17	1/1
	mix	cos	0.21	0.21	0.17	1/1
		mat	0.20	0.20	0.17	1/1

Table 43: Experiment 1: #Nouns (out of 29) $>$ MAJ

	COS	MATCH
SOC	6	7
BI	5	3
UNI	7	8

Table 44: Experiment 1: #Verbs (out of 28) $>$ MAJ

	COS	MATCH
SOC	11	6
BI	5	5
UNI	13	9

Table 45: Experiment 1: #ADJ (out of 15) $>$ MAJ

	COS	MATCH
SOC	1	1
BI	0	0
UNI	1	0

Table 46: Experiment 1: Performance of Second Order Co-occurrences

	COS	MATCH
N	6	7
V	11	6
A	1	1

Table 47: Experiment 1: Performance of Bigrams

	COS	MATCH
N	5	3
V	5	5
A	0	0

Table 48: Experiment 1: Performance of Unigrams

	COS	MATCH
N	7	8
V	13	9
A	1	0

Table 49: Experiment 1: Performance of Cosine Coefficient

	SOC	BI	UNI
N	6	5	7
V	11	5	13
A	1	0	1

Table 50: Experiment 1: Performance of Matching Coefficient

	SOC	BI	UNI
N	7	3	8
V	6	5	9
A	1	0	0

results, while bigrams didn't do as well as was expected. However, we realize that most of the SENSEVAL-2 words have about 100-200 total training instances, which is a fairly small sample of text from which to learn word usages.

Simply put, smaller data leads to smaller number of bigrams. Moreover, bigram feature matching is more demanding than single word matching since it requires two words be matched in a specific order within a small window. Hence, the context vectors based on a smaller set of bigram features are quite sparse and do not provide sufficient information about the target word. The poor performance of bigrams as shown by these experiments suggests that these are not suitable features when the available training data is small in quantity, as is the case of the SENSEVAL-2 words. This result motivates us to consider either improving our context representations or using larger collections of text for training, issues that we explore in subsequent experiments.

Tables 49 and 50 compare the performance of two similarity measures: cosine and match coefficient. Specifically, each value in these tables shows the number of words from a part of speech (as indicated by the row label) that performed better than the majority classifier when using a particular feature type (as shown by the column label) for the selected measure of similarity. Overall, if we look at the total of all values in Tables

49 and 50, we see that cosine performed better than match. This is expected, as the cosine measure takes into account the lengths of the vectors, rather than simply counting the number of matching features.

We also note that the precision and recall of the clustering of the LINE data is generally better than that of the majority sense regardless of the features or measures employed. We believe this is because the number of training instances for the LINE data is significantly higher (1200) than that of the SENSEVAL-2 words. The number and quality of features identified improves considerably with an increase in the amount of training data, making the amount of training data available for feature identification critically important. This motivates us to consider augmenting the training data for SENSEVAL-2 words by collecting instances of these words from the World Wide Web or other larger text collections like the New York Times Newswire corpus or the British National Corpus.

4.2 Experiment 2: First and Second Order Context Representations

In Experiment 1 we represent instances of words using first order context representations, and cluster them using the agglomerative UPGMA algorithm in similarity space. In those experiments, we observed that no matter what feature type we use, the first order context representations are always very sparse due to the fact that each feature contributing to the context vector must occur in that context. Also, there is an additional level of feature matching done by our similarity measure which looks for matching features among these vectors. So, the methodology that we used in the previous experiments not only requires that a feature found in the training is matched exactly in the test, but also expects two test contexts to have matching features to get a similarity score.

In Experiment 1 we observed that this double feature matching (once while finding the context representations and again while computing the similarity scores) ultimately provides very little information to the clustering algorithm, which simply gets a very sparse similarity matrix with very low similarity scores among most of the pairs of contexts.

These findings motivated us to try a better context representation and clustering approach that do not rely on literal feature matching between the test contexts. Experiment 2 [2] was designed in response to these

[2] These experiments were originally published in the Proceedings of the Conference on Computational Natural Language Learning (CONLL-2004) [35]

concerns, and it employs a second order context representation with a hybrid clustering algorithm.

Specifically, these experiments make a systematic comparison among the first and second order context representations using two different kinds of features (bigrams and co-occurrences) and two separate clustering approaches. Contexts are first represented in vector space using the first and second order representations. These vectors are directly clustered in vector space using a hybrid clustering method known as Repeated Bisections [41]. Then, we compute pairwise cosine similarities among the context vectors and perform clustering in similarity space using the UPGMA method, as was done for Experiment 1. We also reduce the dimensionality of second order vectors via Singular Value Decomposition (SVD), in order to discover similarities among the contexts that use conceptually related or synonymous features rather than literally matching strings.

4.2.1 Data

In these experiments, we use 24 of the 73 SENSEVAL-2 words and the LINE, HARD and SERVE corpora.

The LINE, HARD and SERVE corpora do not have a standard training–test split, so these were randomly divided into 60–40 training–test splits. Each of these corpora has about 4000 total instances and hence after splitting, we get approximately 2400 training and 1600 test instances.

Like Experiment 1, we decided to remove the low frequency senses of the SENSEVAL-2 words. However, in Experiment 2 we removed the senses that occur in less than 10% of the total instances for the word. This prevents us from knowing the number of senses associated with any given word, which was not the case in Experiment 1, where we knew there were 5 senses per word. After removing the low frequency senses, we selected all the words that are left with more than 90 training instances. This is because words with fewer training examples than this will generally perform quite poorly due to the lack of a sufficient quantity and quality of features. After this filtering we were left with a set of 24 words which includes 14 nouns, 6 adjectives and 4 verbs as shown in Table 51.

In Experiment 2 we also decided to experiment with discriminating senses of multiple target words simultaneously. For this we created data that has instances of multiple words. To do this, we randomly selected five pairs of words from the SENSEVAL-2 data and combined the training and test instances separately for each pair of words. This gave us a mix training and test sample that includes instances of two different words. After mixing, we applied the 10% sense filter and removed the training and test instances whose correct sense has frequency below 10% of the new mixed sample.

The evaluation of the mixed data is done as if we clustered instances of a single word that has as many senses as the sum of the number of senses of the two mixed words. We do not expect to see two big coarse grained clusters, each containing all instances of a single word, but rather we expect to obtain as many clusters as the total number of senses in the mixed data.

We believed that this mix-word data could be either challenging or easy for discrimination, depending on the degree to which the two mixed words are related. If they share some meanings, then our algorithm might group together their instances, while if the words are unrelated then making coarse grained distinctions between their senses should not be hard. As such, these mix-words provide data with both fine and coarse sense granularities.

Tables 51 and 52 show all the words that were used in Experiment 2, along with their parts of speech. Thereafter we show the number of training (TRN) and test instances (TST) that remain after filtering, and the number of senses found in the test data (S), for each word. We also show the percentage of the majority sense in the test data (MAJ).

In Experiment 2, we set the upper limit on the number of clusters to be discovered to 7. As can be seen from column S in Tables 51 and 52, most of the words have 2 to 4 senses on average. Hence, by creating more clusters, we can test our hypothesis that our clustering approach discovers approximately same number of clusters as senses for that word. We detect the significant clusters by ignoring (i.e., throwing out) clusters that contain less than 2% of the total instances. The instances in the discarded clusters are counted as unclustered instances and are subtracted from the total number of instances while computing the precision value.

4.2.2 Results

We present the discrimination results for six different configurations of features, context representations and clustering algorithms. These were run on each of the 27 target words, and also on the five mixed words. What follows is a concise description of each configuration.

- PB1 : First order context vectors, using co–occurrence features, are clustered in similarity space using the UPGMA technique.
- PB2 : Same as PB1, except that the first order context vectors are clustered in vector space using Repeated Bisections.
- PB3: Same as PB1, except the first order context vectors used bigram features instead of co–occurrences.

All of the PB experiments use first order context representations that correspond to the approach suggested by Pedersen and Bruce.

- SC1: Second order context vectors of instances were clustered in vector space using the Repeated Bisections technique. The context vectors were created from the word co–occurrence matrix whose dimensions were reduced using SVD.
- SC2: Same as SC1 except that the second order context vectors are converted to a similarity matrix and clustered using the UPGMA method.
- SC3: Same as SC1, except the second order context vectors were created from the bigram matrix.

All of the SC experiments use second order context vectors and hence follow the approach suggested by Schütze.

Experiment PB2 clusters using the Pedersen and Bruce style (first order) context vectors, but with the Schütze like clustering scheme. SC2 tries to see the effect of using the Pedersen and Bruce style clustering method on Schütze style (second order) context vectors. The motivation behind experiments PB3 and SC3 is to evaluate bigram features in both PB and SC style context vectors.

The F–measure associated with the discrimination of each word is shown in Tables 51 and 52. Any score that is significantly greater than the majority sense (according to a paired t–test) is shown in bold face. The italicized entries show the best performance (including that of the majority classifier) for each word.

Table 51: Experiment 2: F-measures - First and Second Order Contexts (SENSEVAL-2)

word.pos	TRN	TST	S	PB1	SC1	PB2	SC2	PB3	SC3	MAJ
art.n	159	83	4	37.97	45.52	45.46	46.15	43.03	***55.34***	46.32
authority.n	168	90	4	**38.15**	**51.25**	**43.93**	***53.01***	**41.86**	34.94	37.76
bar.n	220	119	5	34.63	37.23	**50.66**	40.87	41.05	***58.26***	45.93
channel.n	135	67	6	**40.63**	**37.21**	**40.31**	***41.54***	**36.51**	**39.06**	31.88
child.n	116	62	2	45.04	46.85	51.32	50.00	55.17	53.45	*56.45*
church.n	123	60	2	57.14	49.09	48.21	55.36	52.73	46.43	*59.02*
circuit.n	129	75	8	25.17	***34.72***	**32.17**	**33.33**	27.97	25.35	30.26
day.n	239	128	3	60.48	46.15	55.65	45.76	62.65	55.65	*62.94*
facility.n	110	56	3	40.00	**58.00**	38.09	**58.00**	38.46	***64.76***	48.28
feeling.n	98	45	2	58.23	51.22	52.50	56.10	46.34	53.66	*61.70*
grip.n	94	49	5	45.66	43.01	***58.06***	**53.76**	**49.46**	**49.46**	46.67
material.n	111	65	5	32.79	40.98	41.32	***47.54***	32.79	***47.54***	42.25
mouth.n	106	55	4	**54.90**	**47.53**	***60.78***	43.14	43.14	**47.06**	46.97
post.n	135	72	5	**32.36**	**37.96**	***48.17***	30.88	30.88	**32.36**	32.05
blind.a	97	53	3	53.06	61.18	63.64	58.43	76.29	79.17	*82.46*
cool.a	102	51	5	35.42	39.58	38.71	34.78	33.68	38.71	*42.86*
fine.a	93	59	5	**47.27**	***47.71***	***47.71***	33.93	38.18	***47.71***	41.10
free.a	105	64	3	48.74	**49.54**	**52.54**	***55.46***	45.00	**52.99**	49.23
natural.a	142	75	4	34.72	35.21	33.56	30.99	32.40	***38.03***	35.80
simple.a	126	64	4	38.33	50.00	47.06	38.33	38.33	47.06	*50.75*
begin.v	507	255	3	59.36	40.46	40.40	43.66	***70.12***	42.55	64.31
leave.v	118	54	5	**43.14**	**38.78**	27.73	**40.00**	**46.00**	***53.47***	38.18
live.v	112	59	4	37.83	40.00	48.21	45.45	36.37	41.82	*57.63*
train.v	116	56	5	28.57	**33.96**	28.57	***34.28***	26.67	32.08	33.93

Table 52: Experiment 2: F-measures - First and Second Order Contexts (LINE, HARD, SERVE, and mix)

word.pos	TRN	TST	S	PB1	SC1	PB2	SC2	PB3	SC3	MAJ
line.n	1615	1197	3	***72.67***	26.77	62.00	55.47	68.40	37.97	72.10
hard.a	2365	1592	2	86.75	67.42	41.18	73.22	87.06	63.41	*87.44*
serve.v	2365	1752	4	40.50	33.20	36.82	34.37	***45.66***	31.46	40.53
cool.a-train.v	197	102	8	22.34	**39.00**	**25.25**	**40.61**	22.57	***41.00***	22.86
fine.a-cool.a	185	104	7	**27.86**	**42.36**	**33.83**	***47.72***	**35.00**	**42.05**	24.79
fine.a-grip.n	177	99	7	**36.84**	***49.48***	**33.50**	**45.02**	**31.41**	***49.48***	24.19
leave.v-post.n	204	113	8	**29.36**	***48.18***	**32.11**	**41.44**	**23.85**	**41.82**	21.01
post.n-grip.n	208	117	8	**28.44**	***43.67***	**28.44**	**41.05**	**26.55**	**34.21**	20.90

4.2.3 Analysis

We employ three different types of data in Experiment 2. The SENSEVAL-2 words have a relatively small number of training and test instances (around 50-200). However, the LINE, HARD and SERVE data is much larger, where each contains around 4200 training and test instances combined. Mixed words are unique because they combined the instances of multiple target words and thereby have a larger number of senses to discriminate. Each type of data brings with it unique characteristics, and sheds light on different aspects of our experiments. Hence, we analyze the results of each dataset separately.

SENSEVAL-2 data Table 53 compares PB1 against PB3, and SC1 against SC3, when these methods are used to discriminate the 24 SENSEVAL-2 words. Our objective is to study the effect of using bigram features against co–occurrences in first (PB) and second (SC) order context vectors while using relatively small amounts of training data per word. Note that PB1 and SC1 use co–occurrence features, while PB3 and SC3 rely on bigram features.

This table shows the number of nouns (N), adjectives (A) and verbs (V) where bigrams were more effective than co-occurrences (bigram$>$co-occur), less effective (bigram$<$co-occur), and had no effect (bigram=co-occur).

Table 53 shows that there is no clear advantage to using either bigrams or co–occurrence features in first order context vectors (PB). However, bigram features show clear improvement in the results of second order context vectors (SC).

Our hypothesis is that first order context vectors (PB) represent a small set of bigram features since they are selected from the relatively smaller training data. These features are very sparse, and as such most instances do not share many common features with other instances, making first order clustering difficult.

However, second order context vectors indirectly represent bigram features, and do not require an exact bigram match between vectors in order to establish similarity. The matching is still performed at the single token level. Thus, the poor performance of bigrams in the case of first order context vectors suggests that when dealing with small amounts of data, we need to boost or enrich our bigram feature set by using some other larger training source like a corpus drawn from the Web.

Table 54 shows the results of using the Repeated Bisections algorithm in vector space (PB) against that of using UPGMA method in similarity space. This table shows the number of Nouns, Adjectives and Verbs SENSEVAL-2 words that performed better (rbr>upgma), worse (rbr<upgma), and equal (rbr=upgma) when using Repeated Bisections clustering versus the UPGMA technique, on first (PB) and second (SC) order vectors.

In short, Table 54 compares PB1 against PB2 and SC1 against SC2. From this, we observe that with both first order and second order context vectors, Repeated Bisections is more effective than UPGMA. This suggests that it is better suited to deal with very small amounts of sparse data. This could be because the Repeated Bisections method uses a partitional divisive approach that simply divides the given set of vectors in space, rather than performing detailed pairwise comparisons as done by UPGMA. With sparse vector representations as we get with smaller training, such fine comparisons in similarity space might not be helpful as most of the instances will have low similarity values.

Table 55 summarizes the overall performance of each of these experiments compared with the majority class. This table shows the number of words for which an experiment performed better than the majority class, broken down by part of speech. Note that SC3 and SC1 are most often better than the majority class, followed closely by PB2 and SC2. This suggests that the second order context vectors (SC) have an advantage over the first order vectors for small training data as is found among the 24 SENSEVAL-2

Table 53: Experiment 2: Bigrams vs. Co-occurrences

	N	A	V	
	7	1	2	bigram>co-occur
PB	6	4	2	bigram<co-occur
	1	1	0	bigram=co-occur
	9	3	3	bigram>co-occur
SC	4	1	1	bigram<co-occur
	1	2	0	bigram=co-occur

Table 54: Experiment 2: Repeated Bisections vs. UPGMA

	N	A	V	
PB	9	4	1	rbr>upgma
	4	0	2	rbr<upgma
	1	2	1	rbr=upgma
SC	8	1	3	rbr>upgma
	2	5	0	rbr<upgma
	4	0	1	rbr=upgma

words.

We believe that second order methods work better on smaller amounts of data, in that the feature spaces are quite small, and are not able to support the degree of exact matching of features between instances that first order vectors require. Second order context vectors succeed in such cases because they find indirect second order co–occurrences of feature words and hence describe the context more extensively than the first order representations.

With smaller quantities of data, there is less possibility of finding instances that use exactly the same set of words. Semantically related instances use words that are conceptually the same but perhaps not lexically. Second order context vectors are designed to identify such relationships, in that exact matching is not

Table 55: Experiment 2: All vs. Majority Class

	N	A	V	TOTAL
SC3 > MAJ	8	3	1	12
SC1 > MAJ	6	2	2	10
PB2 > MAJ	7	2	0	9
SC2 > MAJ	6	1	2	9
PB1 > MAJ	4	1	1	6
PB3 > MAJ	3	0	2	5

required, but rather words that occur in similar contexts will have similar vectors.

LINE, HARD and SERVE data The comparatively good performance of PB1 and PB3 in the case of the LINE, HARD and SERVE data (see Table 52) suggests that first order context vectors when clustered with UPGMA perform relatively well on larger samples of data.

Moreover, among the SC experiments on this data, the performance of SC2 is relatively high. This further suggests that UPGMA performs much better than Repeated Bisections with larger amounts of training data.

These observations correspond with the hypothesis drawn from the SENSEVAL-2 results. That is, a large amount of training data will lead to a larger feature space and hence there is a greater chance of matching more features directly in the context of the test instances. Hence, the first order context vectors that rely on the immediate context of the target word succeed as the contexts are more likely to use similar sets of words that in turn are selected from a large feature collection.

Mix-Word Results Nearly all of the experiments carried out with the 6 different methods perform better than the majority sense in the case of the mix-words. This is partially due to the fact that these words have a large number of senses, and therefore have low majority classifiers which set an easy standard to reach. In addition, recall that this data is created by mixing instances of distinct target words, which leads to a subset of coarse grained (distinct) senses within the data that are easier to discover than the senses of a single word.

Table 52 shows that the top 3 experiments for each of the mixed-words are all second order experiments

(SC). We believe that this is due to the sparsity of the feature spaces of this data. Since there are so many different senses, the number of first order features that would be required to correctly discriminate them is very high, leading to better results for second order vectors.

4.2.4 Conclusions

We conclude that for larger amounts of homogeneous data such as the LINE, HARD and SERVE, the first order context vector representation and the UPGMA clustering algorithm are the most effective at word sense discrimination. We believe this is the case because in a large sample of data, it is very likely that the features that occur in the training data will also occur in the test data, making it possible to represent test instances with fairly rich feature sets. When given smaller amounts of data like SENSEVAL-2, second order context vectors and a hybrid clustering method like Repeated Bisections perform better. This occurs because in small and sparse data, direct first order features are seldom observed in both the training and the test data. However, the indirect second order co–occurrence relationships that are captured by these methods provide sufficient information for discrimination to proceed.

4.3 Experiment 3: Local and Global Training

From the results of Experiments 1 and 2, we realized the need for large amounts of training data. Hence, in Experiment 3, we employ a large newswire corpus instead of relying on a smaller volume of available local training data. Unlike the local training data, the newswire text is a running corpus that includes complete news articles, and is not simply a collection of contexts associated with a specific target word.

The goal of Experiment 3 is to test if a large sample of running (or global) text is a better source of training data for sense discrimination than is a smaller sample of local training data as was used in Experiments 1 and 2. In this section, we present a comparison of results obtained in Experiment 2 against those obtained by using the global training data.

4.3.1 Data

In particular, we used the Associated Press Worldstream English Service (APW) newswire as the source of global training data. This was distributed as a part of the English GigaWord corpus by Linguistic Data Consortium (LDC), at the University of Pennsylvania. This data consists of text collected from the APW newswire from November 1994 to June 2002. It contains a total of 1,477,466 articles and 539,665,000 words. Each news article is divided into a number of paragraphs, each of which we treat as a separate context. When counting bigrams or word co-occurrence pairs, we assume that the scope of each context ends at the paragraph boundary and don't consider pairs of words that span across paragraph boundaries as features. Also, due to the large size of this data, we did not use any window for bigram and co-occurrences. Hence, only pairs of words immediately next to each other are considered.

The test instances are made up of the same SENSEVAL-2 words and the LINE, HARD and SERVE corpora as used in Experiment 2. All the preprocessing was done in the exactly same manner as in Experiment 2, in order to allow for the direct comparison of results.

4.3.2 Results

Table 56 shows F-measure values obtained by running the same six configurations of feature, context, and clustering types as used in Experiment 2, which featured local training. As before, the bold face entries show where the results were significantly more accurate than the majority classifier. The maximum value in each row including the majority sense is italicized. Entries marked X indicate that we we were unable to get SC1 and SC2 results on some words, in particular day.n, begin.v and the line.n, hard.a and serve.v. The co-occurrence matrices created for these words were too large (approximately 7000 x 200,000) to carry out Singular Value Decomposition due to the large amount of memory required to perform the computations. In future, we plan to adjust the cutoff values on frequency and statistical measures of association in the hopes of reducing the number of features. Also, we will explore the use of computationally more efficient implementations of SVD.

Table 57 shows a pairwise comparison of results with global and local training for each experiment on each word. A mark of G indicates that the global experiment was better than the corresponding local experiment, while L indicates otherwise. X shows either a tie between global and local results, or cases where we could

Table 56: Experiment 3: F-Measures with Global Training

word.pos	PB1	SC1	PB2	SC2	PB3	SC3	MAJ
art.n	***47.50***	29.42	35.66	39.73	44.17	28.99	46.32
authority.n	34.15	32.00	***41.13***	**39.49**	37.29	35.14	37.76
bar.n	***66.09***	32.86	45.00	**52.81**	**48.31**	31.53	45.93
channel.n	***58.02***	**35.20**	**45.53**	**42.75**	**51.13**	**33.07**	31.88
child.n	54.94	32.18	35.00	49.09	52.99	33.73	*56.45*
church.n	55.91	32.50	40.00	***62.86***	54.87	33.73	59.02
circuit.n	**40.00**	**40.00**	***48.00***	***48.00***	**34.67**	**38.67**	30.26
day.n	44.92	X	31.70	X	52.59	29.45	*62.94*
facility.n	39.13	35.95	45.78	33.34	40.74	36.15	*48.28*
feeling.n	54.05	38.10	27.59	50.66	56.10	32.78	*61.70*
grip.n	***58.33***	40.00	30.59	**53.19**	44.21	34.48	46.67
material.n	36.67	33.62	34.23	34.92	35.94	39.29	*42.25*
mouth.n	46.46	36.36	35.95	39.59	***48.60***	35.16	46.97
post.n	***47.82***	**34.92**	**43.90**	32.62	**36.62**	**40.32**	32.05
blind.a	41.86	29.41	30.98	51.95	49.48	31.88	*82.46*
cool.a	42.56	36.95	42.69	***48.49***	43.30	29.55	42.86
fine.a	**44.04**	30.48	33.66	41.74	***45.22***	33.01	41.10
free.a	48.74	36.17	28.57	43.86	47.54	39.13	*49.23*
natural.a	34.78	35.00	***44.83***	35.04	33.33	36.80	35.80
simple.a	45.90	34.61	33.01	41.02	43.20	33.01	*50.75*
begin.v	53.44	X	31.89	X	***65.62***	36.51	64.31
leave.v	**43.39**	34.41	30.93	**39.62**	***47.17***	39.13	38.18
live.v	38.53	35.05	36.56	33.65	35.09	36.95	*57.63*
train.v	**37.38**	**41.66**	***42.86***	**39.25**	30.91	**37.11**	33.93
line-n	***74.27***	X	51.00	X	71.28	43.75	72.10
hard-a	86.87	X	35.12	X	74.23	45.64	*87.44*
serve-v	39.58	X	***42.14***	X	36.79	35.18	40.53

Table 57: Experiment 3: Comparing Global and Local Training

word.pos	PB1	SC1	PB2	SC2	PB3	SC3	Best
art.n	G	L	L	L	G	L	L
authority.n	L	L	L	L	L	X	L
bar.n	G	L	L	G	G	L	G
channel.n	G	L	G	G	G	L	G
child.n	G	L	L	L	L	L	L
church.n	L	L	L	G	G	L	G
circuit.n	G	G	G	G	G	G	G
day.n	L	X	L	X	L	L	L
facility.n	L	L	G	L	G	L	L
feeling.n	L	L	L	L	G	L	L
grip.n	G	L	L	X	L	L	X
material.n	G	L	L	L	G	L	L
mouth.n	L	L	L	L	G	L	L
post.n	G	L	L	G	G	G	L
blind.a	L	L	L	L	L	L	L
cool.a	G	L	G	G	G	L	G
fine.a	L	L	L	G	G	L	L
free.a	X	L	L	L	G	L	L
natural.a	X	X	G	G	X	L	G
simple.a	G	L	L	G	G	L	L
begin.v	L	X	L	X	L	L	L
leave.v	X	L	G	X	G	L	L
live.v	X	L	L	L	L	L	L
train.v	G	G	G	G	G	G	G
line.n	G	X	L	X	G	G	G
hard.a	X	X	L	X	L	L	L
serve.v	L	X	G	X	L	G	L

Table 58: Experiment 3: Summarizing Global vs. Local Comparisons

	G	L	X
PB1	12	10	5
PB2	8	19	0
PB3	17	9	1
SC1	2	19	6
SC2	10	10	7
SC3	5	12	1

not compute the global results as indicated in Table 56.

The last column of table 57 compares the best result obtained with global training against the best result obtained with local. The G entry indicates that for that particular word, the highest performance obtained with global training was greater than the highest performance obtained with local training, among the six variations we attempted. L shows otherwise, meaning, the best result of local was better than the best global. X shows the tie between the best global and local. As the table suggests there are very few words on which global data improved the performance. We notice that most of these words (except *line*) are among the words that have the least majority sense frequency combined with a larger number of senses. Note that, this combination of large number of senses and lower majority leads to smaller set of instances using each sense, which in turn leads to overall poor quality feature set that doesn't have sufficient features for any sense. The noun *circuit* is an extreme case with maximum number of senses (8) (even after filtering) and clearly shows quite a lot improvement in all the experiments with global training. This suggests that with large number of senses and lower majority, a larger volume of global data provides better features than smaller amounts of local data.

Table 58 summarizes the information in Table 57 by counting the total number of words for each of the experimental configurations on which the specific type of training was most successful. Specifically, the rows represent the six configurations: PB1, SC1, PB2, SC2, PB3, and SC3. The values in column G indicate the total number of words for which global training was better for the experiment indicated by the corresponding row label. Similarly, the values in column L indicate the number of words for which local training was better and X indicate the ties.

This table shows that global training only improved the performance of configurations PB1 and PB3, both of which used the first order contexts and UPGMA clustering. As discussed in the previous sections, both the settings are especially challenging for smaller amount of training data due to the combination of smaller feature sets, sparse first order context vectors and rigorous comparisons done by UPGMA in similarity space. Global training in this case provides better features (and hence context representations) compared to smaller quantity of local training. However, overall comparison shows that local training even though was employed in smaller amount proves better than larger global training for most of the words and most of the other configurations.

4.4 Experiment 4: Augmenting Training Data with Dictionary Content

In all the experiments thus far, we took a knowledge–lean approach that uses no additional information other than what is present in the raw text. As we noticed in Experiments 1 and 2, this approach doesn't succeed very well when there are small quantities of training data. In Experiment 3, we tried to overcome this limitation by using a large amount of newspaper text as training corpora. However, we found that the performance obtained with global training was no better than local training. Hence, in Experiment 4, we decided to take a more knowledge intensive approach, and utilize the content of an electronic dictionary to improve the quality of our training data.

Recall that second order context vectors represent contexts by an average of the feature vectors of words that appear in that context. A feature vector (like a co-occurrence or bigram vector) is assumed to provide information about the meaning of the corresponding feature word, in terms of the words that often co-occur with it in the given training data. If the training data is limited in size, as was the case in Experiments 1 and 2, the feature vectors will be represented in fewer dimensions and would not have sufficient information to convey the meaning of that word, or in turn, the meaning of the context in which they occur.

In Experiment 4, we enrich these corpus derived feature co-occurrence vectors by adding words that describe the meaning of this feature word in a dictionary. Thus, we represent each feature word observed in the context of the target word in a test data by a vector of words that are either observed in the context of that word in training or that appear in its dictionary definition.

We hypothesize that for each word there is an associated set of words that a human judge would say are related. The nature of this relationship can vary, but it might exist because the words frequently co–occur, or they have similar meanings, or they are used to define each other.

For example, some of the words that a human might think of when they hear the word *SHELL* are *SEASHORE, AMMUNITION, ARTILLERY, COVERING, GUNS, FIRE, EXPLOSIVE, ENVELOP*, etc. We noticed that these are essentially the words that are found in the dictionary definition of *SHELL*, or they are found in the context of *SHELL* in some text. This observation led to Experiment 4, where we decided to augment the word co-occurrence vectors as derived from the small training data with the words that appear in the dictionary definition (or *gloss*) of that word.

We used WordNet-2.0 [12] in Experiment 4 as the source of glosses, but any other machine readable dictionary would suffice. For each word observed in the context of the target word in the test data, we construct a binary co-occurrence vector that shows the words that co-occur in the context of this word in the training data. We then augment each co-occurrence vector with all the content words that appear in the glosses of various senses of that word.

For example, suppose we observe *BOMB* in the context of our target word *SHELLS* in a test instance. Then, we first create a co-occurrence vector for *BOMB* that shows the words that co-occur with this word in training data. Suppose, we observe words *ATOM, NUCLEAR, BLAST, ATTACK, DAMAGE, KILL* in the context of *BOMB* in our training. Note that, this can be viewed as a purely corpus based co-occurrence vector. Now, we refer to WordNet and see the words that appear in various glosses of the word *BOMB*. Some of such words are *ATTACK, DENOTE, EXPLOSIVE, VESSEL, HEAT*. We refer to this as a gloss vector. In these experiments, we take the union of the corpus derived co-occurrence vector with the gloss vector as:

ATOM, NUCLEAR, BLAST, ATTACK, DAMAGE, KILL, DENOTE, EXPLOSIVE, VESSEL, HEAT.

In summary, if a word is observed in the context of the target word in a test instance, and appears in both training and WordNet, then its feature vector is a union of its gloss vector and corpus–derived co-occurrence vector. If the word doesn't appear in the training data but appears in WordNet, its feature vector is same as its gloss vector as derived from WordNet. On the other hand, if the word appears in training but not in WordNet, its vector is same as the corpus–derived co-occurrence vector. And finally, if the word doesn't appear in either training or WordNet, there will be no vector associated with it. And it will contribute no information to the context vector of an instance in which it appears. In short, a context vector is simply an addition (binary OR function) of all feature vectors of content words that appear in the context.

Since the gloss augmentation to the corpus derived co-occurrence vectors leads to significant growth in the dimensionality of the feature space, we perform SVD to reduce the size of the feature space. In this set of experiments, we reduce the feature space to 2% of its size after gloss additions.

The context vectors are then clustered using UPGMA which showed better performance over Repeated Bisections for a larger amount of training data in our previous set of experiments.

4.4.1 Data

As in previous experiments, here we also used the SENSEVAL-2 corpus, and the LINE, HARD and SERVE corpora.

Low frequency senses from the SENSEVAL-2 words were filtered using a 5 percent sense-filter. In these experiments, we did not select any subset of SENSEVAL-2 words based on the size of training as the training data for all words will be augmented with WordNet glosses. however, we did remove the word *ferret* since it only has three test and training instances in total. Thus, we used 72 of the 73 SENSEVAL-2 words in these experiments.

As was the case in previous experiments, the LINE, HARD and SERVE corpora were randomly split into 60–40 training–test partitions. Thus, for all the words the training data consists of the local contexts around a specific target word as was used in Experiments 1 and 2. Finally, in this experiment we found 10 clusters for each word.

4.4.2 Results and Analysis

Table 59 shows the F–measure of word sense discrimination attained for each word, with *(F-gl)* and without *(F-nogl)* gloss augmentation. Entries in bold type show the experiments where gloss augmented feature vectors resulted in significantly better performance than using feature vectors derived strictly from training data.

Out of the 72 SENSEVAL-2 words, a total of 43 showed improved F-measures using gloss augmented feature vectors. There were seven words that showed no significant change, which suggests that only 22 words showed drop in the F-measure on gloss augmentation. In addition, all of these 43 words also showed improved recall when using gloss augmented feature vectors, which shows that the number of instances correctly clustered was increased due to the use of the gloss augmentation.

Further examination showed that not all of the 43 words that improved overall showed a corresponding increase in their precision. This further indicates that the gloss augmentation not only increased the number of instances correctly clustered but also increased the total number of instances attempted by the algorithm. This is because the rise in the total number of instances correctly clustered as indicated by the improved recall, was accompanied by a rise in the total number of instances attempted, resulting in relatively steady precision.

Our hypothesis is that the sparsity in the feature vectors without gloss augmentation left large number of instances unclustered due to very low levels of similarity with any of the other instances. We believe that gloss augmentation increases the likelihood of discriminating instances that have a very distinct set of features that may not be shared by other instances. Thus, the gloss augmentation allowed for a certain amount of standardization in the feature vectors, which raised the number of instances that were successfully clustered.

However, the results for *line*, *hard* and *serve* do not show any clear improvement when using gloss augmented feature vectors. We believe that this is due to the fact that most of the words that occur in the dictionary glosses of these words have already occurred in these larger samples of training data, so the gloss information is essentially redundant. Thus, we believe that gloss augmented feature vectors are particularly useful for situations where unsupervised discrimination must be performed using smaller samples of training data.

Table 59: Experiment 4: F–measures with *(F-gl)* and without *(F-nogl)* gloss augmentation

word	F-nogl	F-gl	word	F-nogl	F-gl	word	F-nogl	F-gl
art.n	40.00	**50.95**	authority.n	49.70	40.00	bar.n	54.39	50.44
begin.v	49.69	**59.88**	blind.a	32.43	**45.00**	bum.n	60.32	36.36
call.v	35.44	**37.11**	carry.v	44.74	40.97	chair.n	48.00	**71.03**
channel.n	45.16	32.81	child.n	56.86	50.91	church.n	41.76	**54 .37**
circuit.n	42.46	34.24	collaborate.v	40.00	**59.09**	colourless.a	56.00	**58.62**
cool.a	31.32	**35.56**	day.n	44.15	**65.31**	detention.n	62.22	42.55
develop.v	34.55	**39.64**	draw.v	41.86	**52.38**	dress.v	37.89	37.50
drift.v	39.29	**46.43**	drive.v	45.61	**54.54**	dyke.n	48.78	**60.00**
face.v	41.79	**77.01**	facility.n	43.90	**46.00**	faithful.a	4 2.42	42.42
fatigue.n	49.18	**64.79**	feeling.n	33.90	**46.58**	find.v	30. 23	**41.86**
fine.a	41.51	**48.21**	fit.a	40.91	40.91	free.a	45.61	**47. 79**
graceful.a	38.89	38.89	green.a	56.21	55.07	grip.n	41.46	**53.33**
hearth.n	57.70	44.90	holiday.n	37.74	**44.89**	keep.v	35.82	**67.50**
lady.n	37.34	**54.54**	leave.v	50.98	39.60	live.v	36.36	31.77
local.a	44.07	41.94	match.v	41.27	**52.94**	material.n	38.71	**41.60**
mouth.n	33.71	**39.21**	nation.n	59.26	**76.67**	natural.a	33.07	**34.78**
nature.n	36.84	33.73	oblique.a	40.00	**54.55**	play.v	48.72	37.33
post.n	47.70	39.39	pull.v	45.28	44.44	replace.v	38.24	**52.38**
restraint.n	40.54	35.90	see.v	33.34	**34.70**	sense.n	32.19	**39.08**
serve.v	50.64	45.98	simple.a	33.96	**47.06**	solemn.a	25.00	**47.06**
spade.n	44.90	**48.14**	stress.n	42.86	36.07	strike.v	37.50	**40.62**
train.v	41.13	41.13	treat.v	47.76	47.37	turn.v	40.00	34.62
use.v	31.20	**62.12**	vital.a	5.56	5.56	wander.v	30.13	**56.41**
wash.v	66.67	60.00	work.v	39.21	**49.18**	yew.n	56.41	**68.19**
line.n	43.13	43.04	hard.a	67.25	67.09	serve.v	38.54	36.60

5 Conclusions

One of the main objectives of this thesis was to determine the impact on word sense discrimination of different feature types, context representations and clustering methods.

This thesis shows that there is no unique configuration of choices that gives the best results on all datasets. However, we do make specific recommendations for carrying out discrimination, based on the nature and volume of data used for training and clustering.

The following sections summarize what settings achieve the best discrimination under the different scenarios we considered.

5.1 Smaller Datasets

We observed that smaller amounts of training data lead to smaller feature sets. First order context representations based on smaller feature sets tend to be very sparse since they are based strictly on features that appear in the contexts being discriminated. Among the various features we employed, we observed that single token features (e.g., unigrams, co-occurrences) perform better with smaller datasets than multi-token features (e.g., bigrams), which are less likely to occur in a context being discriminated.

Feature sets that are both small and sparse make the first order features a limited presentation of contexts that does not convey much information about the meaning of the target word. Similarity measures add an extra level of feature matching by seeking matching features among the contexts, which makes the similarity space representations even more sparse than the corresponding vector space representations. Given such very sparse and limited information, the agglomerative clustering methods that rely on rigorous pairwise comparisons among the contexts did not fare very well.

On the other hand, second order context representations introduce additional information into the context vectors by adding feature vectors of contextual words. The words representing dimensions of the context vectors are not required to appear in the contexts of the target word but do appear in the contexts of the contextual feature words. Hence, the 2nd order contexts tend to be more dense, informative and extensive representations of contexts even with smaller feature sets. We also noticed that the indirect representation of bigrams in second order contexts proves more effective than matching two word sequences as done by the first order contexts. We also observed that the dimensionality reduction helps in identifying contexts that use similar or related sets of features rather than literally matching text strings.

Considering the smaller number of dimensions in a feature space that results from smaller training data, when combined with the inherent sparsity in natural language text, we showed that the partitional clustering approach that directly clusters contexts in vector space is more effective than agglomerative clustering in similarity space.

In summary, with a smaller volume of training and test data, the second order context representations using either bigram or co-occurrence features, when clustered using vector space hybrid methods like repeated bisections, tend to achieve better discrimination.

5.2 Larger Datasets

Larger local training data that consists of contexts around a specific target word seems to provide overall better quality and quantity of features. As the improved feature selection increases the likelihood of directly matching these features in the contexts of test instances, the first order context representations get richer and provide some substantial information about the features directly observed near the target word in the contexts. This in turns improves the chances of detecting direct similarities among the contexts in terms of the shared features. Under these conditions, we noticed that the detailed pairwise comparisons done by the agglomerative clustering algorithm result in better discrimination than a hybrid clustering approach such as Repeated Bisections.

But, we noticed some adverse effects on second order contexts with the additional training data. We believe this is because they include some extra information into the contexts by adding co-occurrences of contextual features. This technique could in fact introduce significant amounts of noise that in turn can obscure the fine level distinctions between the contexts. In other words, when the first order contexts already have sufficient information to identify similar contexts, the added information about co-occurrences of feature words is not necessary and in fact can confuse the clustering algorithm.

5.3 Global Training with Large Generic Text

Our experiments with large global training showed that it doesn't perform as well as local training done with a smaller quantity of data. Global training only seems to boost the performance of the first order contexts, especially those that use bigram features collected from a smaller sample of local training. In such cases, we noticed that the larger volume of data gives better features that improve the first order representations. It also helped in discriminating instances of words that have a large number of senses. In short, we noticed that global training can only outperform the discrimination results obtained with insufficient local training. In the majority of cases, however, even smaller local training proved better than larger global training.

5.4 Comparisons Against a Knowledge–Intensive Approach

We compared our results obtained with a knowledge–lean approach against those obtained with a more knowledge–intensive method that incorporated dictionary definitions of feature words into contexts. For each word observed in the context of a test instance, we created a feature vector of words that co-occur with that feature word in the training data. we then augmented each feature vector with words that appear in the WordNet gloss of the feature word. 2nd order context vectors were then computed by averaging such gloss augmented feature vectors of words found in the contexts.

We found that this gloss augmentation only proved better in cases when smaller training data was used for creating feature vectors, and didn't prove useful in combination with larger data. This suggests that the co-occurrence behavior of words as learned from a large raw text certainly has the potential to outperform knowledge–intensive methods.

6 Related Work

There is a long history of research in supervised approaches to word sense *disambiguation*. Typically these approaches train a model by presenting it with some number of manually created sense tagged examples for a particular word (e.g., [5], [7],[26], [29], [22]). After training, these models are able to assign one of a predefined set of meanings to newly encountered instances of a word.

However, word sense discrimination is a different problem. Rather than trying to assign an instance of a word to one of a set of possible meanings, it seeks to group together instances of words that are used in similar contexts. The motivation behind taking this approach is that a predefined set of meanings (as provided by a dictionary or similar resource) is often too inflexible and limited to account for word usages in actual text. In addition, sense tagged text only exists in small quantities and is expensive to create.

Methods of discrimination that discover meanings of words from raw text avoid both of those limitations, and have become more widely studied as the amount and variety of online text continues to increase. Thus, word sense discrimination lends itself to unsupervised *knowledge lean* approaches, while word sense disambiguation tends to be pursued using harder to obtain resources such as sense tagged text.

The following discussion pays particular attention to earlier discrimination work by Schütze and by Pedersen and Bruce. The combination of these two bodies of research serves as the foundation of this thesis. There is also discussion of the related problems of finding sets of words with the same or similar meaning, and bootstrapping approaches that use very small amounts of training data to initiate a fully automatic process.

6.1 Word Sense Discrimination

This thesis explores the effect of vector versus similarity space representations, as well as first order versus second order features. These issues were raised in two different bodies of previous work that provide a starting point for this thesis. Pedersen and Bruce ([30], [31]) explored the use of similarity spaces and first order features, while Schütze ([38], [39]) developed an approach based on vector spaces and second order features. In this thesis, we seek to compare, contrast, and extend these methods.

Pedersen and Bruce compare various hierarchical agglomerative clustering algorithms, and recommend the use of McQuitty's Similarity Analysis [24]. In fact, McQuitty's method is a form of hierarchical agglomerative clustering that uses the average link criteria function. It starts by assuming that each instance is a separate cluster. It merges together the pair of clusters that have the highest average similarity value. This continues until a specified number of clusters is found, or until the similarity measure between every pair of clusters is less than a predefined cutoff.

Pedersen and Bruce use a relatively small number of first order features. They create a dis–similarity matrix by using the matching coefficient as their criterion. Rather than using the number of features that match, they used the number of features that didn’t match, and treat this as a distance measure. The context of a target word is represented using localized first order features such as collocations and part of speech tags that occur within one or two positions of the target word.

By way of contrast, Schütze [39] performs discrimination through the use of a vector based representation. In fact, he employs two different vectors: the first is a word vector that is based on co–occurrence counts from a separate training corpus. Each word in this corpus is represented by a vector made up of the words with which it co-occurs. These vectors are then reduced via Singular Value Decomposition. Then, each instance in a test or evaluation corpus is represented by a vector that is the average of all the vectors of all the words that make up that instance. Discrimination is carried out by clustering instance vectors using a hybrid clustering method that integrates the EM Algorithm with agglomerative clustering.

Below we summarize some of the significant differences in the approaches suggested by Pedersen and Bruce and by Schütze. In this thesis we carry out experiments that isolate some of these differences, in order to determine which techniques are most effective for word sense discrimination.

Context Representation Pedersen and Bruce represent the context of each test instance as a vector of features that directly occur near the target word in that instance. We refer to this representation as the first order context vector. Schütze, by contrast, uses the second order context representation that averages the first order context vectors of individual features that occur near the target word in the instance. Thus, Schütze represents each feature as a vector of words that occur in its context and then computes the context of the target word by adding the feature vectors of significant content words that occur near the target word in that context.

Features Pedersen and Bruce use a small number of local features that include co–occurrence and part of speech information near the target word. They select features from the same test data that is being discriminated, which is a common practice in clustering in general. Schütze represents contexts in a high dimensional feature space that is created using a separate large corpus (referred to as the training corpus). He selects features based on their frequency counts or log-likelihood ratios in this corpus.

In this thesis, we adopt Schütze's approach and select features from a separate corpus of training data, in part because the number of test instances may be relatively small and may not be suitable for selecting a good feature set. In addition, this makes it possible to explore variations in the training data while maintaining a consistent test set. Since the training data used in unsupervised clustering does not need to be sense tagged, in future work we plan to develop methods of collecting very large amounts of raw corpora from the Web and other online sources and use it to extract features.

Schütze represents each feature as a vector of words that co–occur with that feature in the training data. These feature vectors are in fact the first order context vectors of the feature words (and not target word). The words that co–occur with the feature words form the dimensions of the feature space. Schütze reduces the dimensionality of this feature space using Singular Value Decomposition, which is also employed by related techniques such as Latent Semantic Indexing [10] and Latent Semantic Analysis [19]. SVD has the effect of converting a word level feature space into a concept level semantic space that smoothes the fine distinctions between features that represent similar concepts.

Clustering Space Pedersen and Bruce represent instances in a (dis)similarity space where each instance can be seen as a point and the distance between any two points is a function of their mutual (dis)similarities. The (dis)similarity matrix showing the pair-wise (dis)similarities among the instances is given as the input to the agglomerative clustering algorithm. The context group discrimination method used by Schütze, on the other hand, operates on the vector representations of instances and thus works in vector space. Also he employs a hybrid clustering approach which uses both an agglomerative and the Estimation Maximization (EM) algorithm.

While the focus of this thesis has been on Pedersen and Bruce, and on Schütze, there have been other approaches to purely unsupervised word sense discrimination. For example, [13] describe a method for discriminating among verb senses based on determining which nouns co–occur with the target verb. Collocations are extracted which are indicative of the sense of a verb based on a similarity measure they derive.

6.2 Finding Sets of Related Words

Finding sets of related words is a close cousin to the problem of word sense discrimination. The objective is to identify words such as *gun* and *pistol* that are often used in the same context, and have essentially the same meaning. While this is not identical to word sense discrimination, the fact that it is based on finding words that occur in similar contexts makes it very closely related.

Some work on this problem has focused on finding distributional regularities among word occurrences in raw corpora, and grouping together those words that occur in similar contexts. These approaches are based purely on lexical information, and well known examples include [6] and [32]. This style of approach is most closely related to our own, in that only information from raw corpora is employed in making these distinctions.

Other approaches incorporate syntactic information in the form of part of speech tags or partial parses, and identify words that occur in similar contexts based on a combination of textual and syntactic information. Examples include [14], [23], [8], and [28]. The latter two approaches go beyond simply identifying sets of related words and attempt to group those sets into a hierarchy of concepts, where more specific concepts are a form of the more general concepts.

Finally, there are approaches that are initialized with a few seed concepts, and find those words that are related to the given seeds (e.g., [37], [36], [33]). For example, *gun* might be given as a seed, and the method would find *pistol*, *flintlock*, and *artillery*. In general these approaches utilize a combination of lexical and syntactic information.

6.3 Bootstrapping Approaches

There is also research at the intersection of supervised and unsupervised methods. In fact this is more closely related to supervised learning, since the objective remains to create a classifier which will assign an instance of a word to one of a predefined set of possibilities. However, these methods are often minimally supervised, and use a small amount of training data in order to automatically create more training data, or in other words *bootstrap* a large sample of training data from a much smaller sample of sense–tagged data.

The best known example of such an approach is [43], who describes a method that automatically identifies collocations that are indicative of the sense of a word, and uses those to iteratively label more examples. For example, in a large corpus that contains instances of *plant*, it might be that the collocations *manufacturing plant* and *flowering plant* would be identified using standard statistical techniques. A human judge could easily determine that these are associated with two distinct senses of *plant*, and easily label all occurrences of *plant* in each collocation in the corpus with the appropriate sense. Then these examples could be fed back into the learning algorithm, and the resulting model could be used to create more training examples.

7 Future Work

There are many issues that arose during this thesis that suggest future directions for research. These include ideas to improve existing techniques, as well as some new variations that might lead to better discrimination. In addition, we have realized that our methodology is suited to a broad range of problems that extends well beyond word sense discrimination. What follows are our plans for future work.

Training The rather limited effectiveness of global training suggests that we need to revise our feature selection and dimensionality reduction strategies so as to avoid the noise that comes from global data. Also, it encourages us to devise techniques for collecting larger amounts of local training data. In the near future, we will try to use the local contexts around a specific target word for training data as collected from some large text collection like the English Gigaword Corpus, the British National Corpus or the World Wide Web.

Since pure global training seems to have limitations, we will also try to combine it with available smaller local training to determine if that proves better than pure local or pure global training approaches.

In short, the possible variations in training that we would still like to pursue are local training boosted with global training, and enhanced local training by collecting local contexts around the target word from various sources of large text collections.

We also plan to make a more systematic comparison of our knowledge–lean approach against the knowledge–intensive approach, as taken during the gloss experiments. This should help determine if the corpus based approach to discrimination is better than the dictionary based approach under any circumstances.

Features In the future, we will employ richer feature types that use the part of speech or morphology of words and do not just rely on their surface forms. Also, we will explore techniques like stemming and fuzzy feature matching that will hopefully result into richer context representations.

Context Representations This thesis showed that first order context representations are more suitable for large feature sets and larger amounts of training data while second order contexts are more effective when the available training data is small. This gives us an idea of using backoff models that move from first order to second order representation for very sparse contexts. We would also like to combine first and second order context representations in a way that will essentially record both the feature word and its co-occurrences in the context vectors.

SVD We have not yet conducted any experiments to test if the dimensionality reduction has a measurable impact on results. In future, we plan to answer this question by systematically comparing the results of discrimination with and without using SVD. We will also determine the effect of using different reduction factors while carrying out SVD.

Clustering In this thesis, we avoid exactly specifying the number of clusters we expect to find by creating some arbitrarily large number of clusters and then ignoring the extra clusters during evaluation. We understand that this logic results in very low accuracy values if the extra clusters contain a significant number of the instances to be discriminated.

In future, we plan to merge such extra clusters with one of the labelled clusters that shares maximum inter-cluster similarity with that extra cluster. However, this doesn't address the problem of creating the right number of clusters at the first place and without using the knowledge of true classification. One possible solution could be to simply run clustering multiple times, each with a different number of clusters to be found, and then pick the solution that gives the overall maximum intra–cluster similarity and minimum inter–cluster similarity.

Cluster Labeling We have developed a methodology that creates clusters of instances of a target word that refer to the same sense. We currently do not make any attempt to identify which sense each cluster represents. In future, we plan to attach some descriptive labels to the discovered clusters that indicate the sense these clusters represent. Such labels can be created from the features shared by instances in the same cluster and those not shared by instances in other clusters. By comparing such labels with actual dictionary senses of a word, we can determine how well a knowledge–lean approach can perform fully automatic word sense disambiguation without relying on any manually annotated training data or other external knowledge source.

Applications Our focus in this thesis has been on word sense discrimination. However, we have come to realize that the idea of clustering similar text instances could find direct applications in a variety of natural language processing tasks like text summarization, synonymy identification, document clustering and indexing, etc.

There are a number of applications that we would like to explore in the near future.

- Automatic Email Foldering

  In the same way that wwe cluster contexts that contain a specific target word, we can treat each email as a single context and cluster a collection of emails based on the similarity of their contents. In this case there would not be a specific target word, rather an entire email message would be both the target and the context. The development of such a technique will help to automatically organize a large corpus of emails based on their content.

- Name Discrimination

  This is the task of trying to identify the different people associated with the same name. If we think of the actual people with the same name as different meanings of that ambiguous name, we can see that this is essentially the word sense discrimination problem. Given a number of text instances that refer to an ambiguous name, our algorithm will try to automatically group together all instances that refer to the same person. For example, if we search Google for ”Ted Pedersen”, we hit many pages some of which refer to Prof. Ted Pedersen at University of Minnesota, Duluth, while, others to the author of children’s books like ”Internet for Kids”, ”Gipsy World”, ”Ghost on the Net”, etc. Given these two different personalities of the same name, we can apply our discrimination techniques to identify which pages refer to the same person.

- Ontology Acquisition

  Our current strategy is to cluster instances of the same word in order to identify different senses of that word. The other possibility would be to cluster instances of multiple words in order to find the sets of related words similar to the work done by [23], [28]. We realize that, a standard hierarchical clustering will give us an hierarchy of word clusters that shows the different clusters to which a word belongs at various levels. Such an hierarchy can be in fact viewed as an ontology constructed from purely raw text. In future, we plan to analyze and evaluate such automatically constructed ontology and find its applications in some standard NLP problems.

- Synonymy Identification

  This is the problem of identifying words that are synonyms. If our method is used to cluster instances of different words that are synonyms, we hope to see a cluster that groups the instances of two different words that use the same sense. In short, by clustering different contexts in which words *bat* and *club* are used, we should be able to get a single cluster containing all instances of these words that refer to the sense of *stick used for hitting*.

References

[1] G. Augustson and J. Minker. An analysis of some graph theoretical cluster techniques. *ACM*, 17(4):571–588, October 1970.

[2] M. Berry, T. Do, G. O'Brien, V. Krishna, and S. Varadhan. SVDPACK (version 1.0) user's guide. Technical Report CS-93-194, University of Tennessee at Knoxville, Computer Science Department, April 1993.

[3] M.W. Berry, S. Dumais, and G. O'Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, 1995.

[4] M.W. Berry, S. Dumais, and A. Shippy. A case study of latent semantic indexing. Technical Report CS-95-271, University of Tennessee at Knoxville, Computer Science Department, January 1995.

[5] E. Black. An experiment in computational discrimination of English word senses. *IBM Journal of Research and Development*, 32(2):185–194, 1988.

[6] P. Brown, P. deSouza, R. Mercer, T. Watson, V. Della Pietra, and J. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4), 1992.

[7] R. Bruce and J. Wiebe. Word-sense disambiguation using decomposable models. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 139–146, 1994.

[8] S. Caraballo. Automatic acquisition of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 120–126, 1999.

[9] D. Cutting, D. Karger, J. Pedersen, and J. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329, Copenhagen, Denmark, 1992.

[10] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

[11] P. Edmonds and S. Cotton, editors. *Proceedings of the Senseval–2 Workshop*. Association for Computational Linguistics, Toulouse, France, 2001.

[12] C. Fellbaum, editor. *WordNet: An electronic lexical database*. MIT Press, 1998.

[13] F. Fukumoto and Y. Suzuki. Word sense disambiguation in untagged text based on term weight learning. In *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 209–216, Bergen, 1999.

[14] D. Hindle. Noun classification from predicate-argument structures. In *Proceedings of the 28th Meeting of the Association for Computational Linguistics*, pages 268–275, Pittsburgh, PA, 1990.

[15] http://www.altavista.com/.

[16] A. Jain, M. Murthy, and P. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, September 1999.

[17] R. Johnson and D. Wichern. *Applied Multivariate Statistical Analysis*. Prentice–Hall, Inc., Upper Saddle River, NJ, fifth edition, 2002.

[18] H.W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[19] T.K. Landauer, P.W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25:259–284, 1998.

[20] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 16–22, San Diego, CA, 1999.

[21] C. Leacock, G. Towell, and E. Voorhees. Corpus-based statistical sense resolution. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 260–265, March 1993.

[22] K.L. Lee and H.T. Ng. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 41–48, 2002.

[23] D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics*, pages 768–774, Montreal, 1998.

[24] L. McQuitty. Similarity analysis by reciprocal pairs for discrete and continuous data. *Educational and Psychological Measurement*, 26:825–831, 1966.

[25] G.A. Miller and W.G. Charles. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28, 1991.

[26] R. Mooney. Comparative experiments on disambiguating word senses: An illustration of the role of bias in machine learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 82–91, May 1996.

[27] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5:32–38, 1957.

[28] P. Pantel and D. Lin. Discovering word senses from text. In *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining-2002*, 2002.

[29] T. Pedersen. A decision tree of bigrams is an accurate predictor of word sense. In *Proceedings of the Second Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 79–86, Pittsburgh, July 2001.

[30] T. Pedersen and R. Bruce. Distinguishing word senses in untagged text. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 197–207, Providence, RI, August 1997.

[31] T. Pedersen and R. Bruce. Knowledge lean word sense disambiguation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 800–805, Madison, WI, July 1998.

[32] F. Pereira, N. Tishby, and L. Lee. Distributional clustering of English words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 183–190, Columbus, OH, 1993.

[33] W. Philips and E. Riloff. Exploiting strong syntactic heuristics and co-training to learn semantic lexicons. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 125–132, Philadelphia, PA, 2002.

[34] A. Purandare. Discriminating among word senses using McQuitty's similarity analysis. In *Proceedings of the HLT-NAACL 2003 Student Research Workshop*, pages 19–24, Edmonton, Alberta, Canada, May 27 - June 1 2003.

[35] A. Purandare and T. Pedersen. Word sense discrimination by clustering contexts in vector and similarity spaces. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 41–48, Boston, MA, 2004.

[36] E. Riloff and J. Shepherd. A corpus-based bootstrapping algorithm for semi-automated semantic lexicon construction. *Journal of Natural Language Engineering*, 5(2):147–156, 1999.

[37] B. Roark and E. Charniak. Noun-phrase co-occurrence statistics for semi-automatic semantic lexicon construction. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics*, pages 1110–1116, Montreal, 1998.

[38] H. Schütze. Dimensions of meaning. In *Proceedings of Supercomputing '92*, pages 787–796, Minneapolis, MN, 1992.

[39] H. Schütze. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123, 1998.

[40] Lawrence Spence, Arnold Insel, and Stephen Friedberg. *Elementary Linear Algebra: A Matrix Approach*. Prentice–Hall, Inc., Upper Saddle River, NJ, 2000.

[41] M. Steinbach, G. Karypis, and V. Kumar. A comparision of document clustering techniques. In *Proceedings of the Workshop on Text Mining at the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, MA, 2000.

[42] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, November 1993.

[43] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, MA, 1995.

[44] Y. Zhao and G. Karypis. Criteria functions for document clustering: Experiments and analysis. Technical Report 01-040, University of Minnesota, Department of Computer Science, February 2002.

[45] Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the 11th International Conference on Information and Knowledge Management*, pages 515–524, McLean, VA, 2002.

[46] Y. Zhao and G. Karypis. Hierarchical clustering algorithms for document datasets. Technical Report 03–027, University of Minnesota, Department of Computer Science, 2003.