CHAPTER 2

BACKGROUND

In this chapter, we will describe theoretical background that is used in this thesis. In the rest of this chapter, the evaluation criteria of Word Sense Disambiguation are provided.

2.1 Theoretical Background

2.1.1 Vector Space Based Model

With the increasing availability of information in electronic form, it becomes more important and feasible to have automatic methods to retrieve such information. In addition to the conventional method by keywords, many methods, such as full-text search, passage retrieval, contents retrieval. Probably the best known model in Information Retrieval is the vector-based model.

The vector space based model originated by Gerard Salton (Salton and McGill, 1983; Salton, 1989) represents documents as vectors in a vector space. The basic idea is Firstly, to extract unique content-bearing words from the set of documents and treat these words as features and secondly, to represent each document as a vector of certain weighted word frequencies in this feature space. The vector space based model of a text data set can be formed as a *word-by-document matrix* (matrix A) whose rows are words and columns are document vectors. Each entry of matrix A, $A(i, j)$ represents the weighted frequency of term i in document j. Typically, a large number of words exist in even a moderately sized set of documents. Hence, the document vectors are very *high dimensional*. However, typically, most documents contain many fewer words, 1-5% or less, in comparison to the total number of words in the entire document collection. Therefore, the document vectors are very sparse.

2.1.2 Advantages and Disadvantage of Vector Space Based Model

Advantages of Vector Space Based Model

The vector space based model has some desirable features. The semantic content does not reside in the vectors, but rather in the relations between the vectors. Another appealing property of the vector space based model is that the semantic information is extracted automatically, in an unsupervised approach from unstructured text data. It does not involve human interaction. It uses statistical methods in the text data, i.e. usually co-occurrence information, to automatically construct the vectors and the vector spaces. No prior knowledge of the text data is assumed, making the models easy to apply to text data with different topical and structural properties.

The vector space based model is inherently adaptive when applied to new domains, since the dynamics of the semantic will reflect the semantics dynamics of the training data. This means that different domains will produce different semantic spaces, with different semantic relations between different words (Landauer and Dumais, 1997).

The vector space based model does not require and do not provide any answer to the questions what meaning is. The models reflect how words are actually used in a sample of natural language behavior. The only assumption that has to be made is the hypothesis that two words are semantically related if they are used in a similar manner.

Disadvantages of Vector Space Based Model

Although the vector space based model is one promising method for improving the performance of information, the conventional vector space based model uses so many words per vector element that similarity calculation requires much time. Many words per vector element also make the original term by document matrix is very sparse and contains noise data. Moreover, when the query sentence includes only a few words pertaining to the vector elements, the query vector becomes

too sparse to find the relevant documents. In order to resolve these problems, many researches have been conducted. The simplest way is to reduce the vector dimension.

In the case of vector-based model, it has been assumed that the meanings of the words which represent the bases of vector are independent from each other, however, this assumption does not hold in actual documents. Then, in order to reduce the number of dimension, the vector dimension reduction methods such as Latent Semantic Indexing (LSI) method (Deerwester et al., 1990) was proposed where new bases were generated by linear combination of vector bases. The matrix which is transformed by reducing of the vector dimension is dense matrix without noise data.

2.2 Latent Semantic Indexing (LSI)

In the initial vector space, the word-to-document relations contain redundancy, ambiguity, and noise. We want to create a subspace that contains only meaningful semantic associations and is much smaller than the initial space. One method to achieve this is to perform a dimensionality reduction so as to select a semantic subspace that contains essential and meaningful associative relations. In the subspace, redundant information can be grouped together, ambiguity can be coped with by combining with other contextual information, and noise can be partly removed by trimming the least important dimensions of the subspace.

LSI is one such dimension reduction method. It automatically computes a subspace containing meaningful semantic associations which is much smaller than the initial space. This is done through the Singular Value Decomposition (SVD) of the term-document matrix (Strang, 1980).

LSI relies on a Singular Value Decomposition (SVD) of a matrix (word × context) derived from a corpus of natural context that pertains to knowledge in the particular domain of interest (Strang, 1980). SVD is a form of factor analysis and acts as a method for reducing the dimensionality of a feature space without serious loss of specificity. Typically, the word by context matrix is very large and quite often sparse. SVD reduces the number of dimensions without great loss of

descriptiveness. SVD is the underlying operation in a number of applications including statistical principal component analysis (Jolliffe, 1986), text retrieval (Berry et al., 1995) and (Dumais, 1994), pattern recognition and dimensionality reduction (Duda, 1973), and natural language understanding (Landauer, 1997). LSI includes three steps (Deerwester et al., 1990):

Step 1. A large body of text is represented as an occurrence matrix (i × j) in which rows stand for individual word types, columns for meaning bearing passages such as sentences or paragraphs, that is (word × context). Each cell then contains the frequency with which a word occurs in a passage.

Step 2. The matrix is then subject to SVD (Dumais, 1994), (Jolliffe, 1986) and (Strang, 1980).

Step 3. Finally, all but the *d* largest singular values are set to zero. Pre-multiplication of the right-hand matrixes produces a least-squares best approximation to the original matrix given the number of dimensions, *d*, that are retained. The SVD with dimension reduction constitutes a constraint satisfaction induction process in that it predicts the original observations on the basis of linear relations among the abstracted representations of the data that it has retained.

Dimensionality Reduction via SVD

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

$$SVD(A) = UDV' \qquad (2.1)$$

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

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).

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.

When multiplying matrices U, D and V′, the original matrix A is returned. However, the goal of LSI 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 (define k). This is achieved by selecting first k significant singular values from matrix. This has the effect of reducing the dimensionality of matrix A to k dimensions. Figure 2.1 shows how matrices U, D and V′ are truncated to obtain the best approximation of matrix A (A_k) in k dimensions where $k < \text{rank}(A)$.

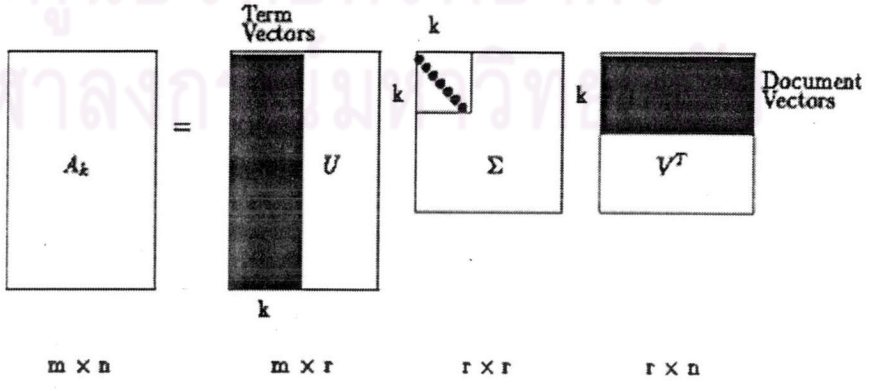

Figure 2.1: Reducing a Matrix to K Dimensions with SVD.

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. The result is that each word is represented as a vector of length k. Performance depends strongly on the choice of the number of dimensions. The optimal number of selecting k is typically around between 200 and 300 and may vary from corpus to corpus, domain to domain. The similarity of any two words, any two text passages, or any word and any text passage, are computed by measures on their vectors. Often the cosine of the contained angle between the vectors in k-space is used as the degree of qualitative similarity of meaning. The length of vectors is also useful as a measure.

The advantages of using LSI for word sense disambiguation is that it is a fully automatic, corpus based statistical procedure that does not require syntactic analysis. LSI uses a fully automatic mathematical/statistical technique to extract and infer semantic relations between the meanings of words from their contextual usage in large collections of natural discourse. The analysis yields a mathematically well-defined representation of a word that can be thought of as a kind of average meaning.

2.3 Data Clustering Methods

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 (Jain et al., 1999).

Typical pattern clustering method can be processed in 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. *Similarity Measurement and Clustering Method:* 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.

2.3.1 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.

Although 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 occur in all documents, it does not 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.

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.

2.3.2 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 occur 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 2.2.

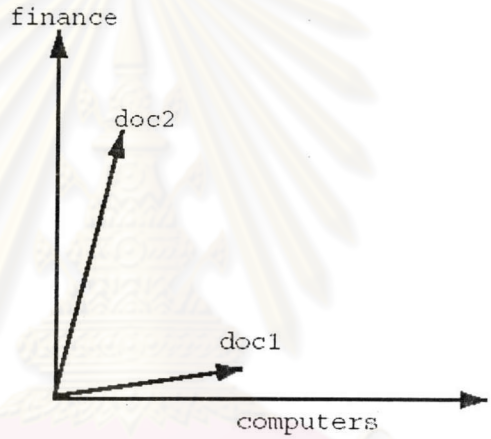

Figure 2.2: Vector Representations of Objects.

In co-ordinate space, document1 can be seen as the point (3, 1), while document2 as the point (2, 4), as shown in Figure 2.3.

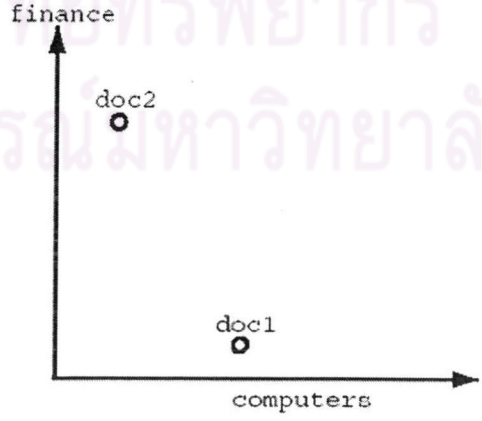

Figure 2.3: Point Representations of Objects.

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 (e.g. firewall, security, recovery, virus, authentication, and encryption).

2.3.3 Similarity Measurement and Clustering Method

2.3.3.1 Similarity Measurement

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 such as Euclidean Distance, Cosine Similarity Coefficient that can be employed. These measures are employed with real–valued feature spaces, and can also be used with integer or binary feature spaces. We briefly review Euclidean Distance, Cosine Similarity Coefficient measures below.

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

$$euclidean\_dist(\vec{x},\vec{y}) = |\vec{x}-\vec{y}| = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2} \qquad (2.2)$$

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.

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. The similarity between two vectors reflects the distributional similarity between the words represented by the vectors. Distributional similarity implies similar values of semantic information. In this thesis, the cosine of the angles between the context vectors is used to calculate the correlation between words. The cosine measure for two vectors $\vec{x}$ and $\vec{y}$ can be calculated as follows:

$$\cos(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{|\vec{x}|\cdot|\vec{y}|} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\cdot\sqrt{\sum_i y_i^2}} \qquad (2.3)$$

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.

The reason for selecting the cosine measure rather than Euclidean distance is that this similarity measure seems less effective for word sense disambiguation, because Euclidean distance is sensitive to vector length that is usually proportional to the frequency of collocating words. In other words, Euclidean distance usually assigns lower similarities to frequently appearing word senses.

2.3.3.2 Clustering Methods

Clustering methods 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 methods* 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 methods can also be classified into three main categories based on the object representation they use. *Vector space* methods directly cluster feature vectors. *Graph based* methods represent objects as graphs, and use graph partitioning techniques to cluster them. This thesis only concerns vector methods, but not graph based approaches.

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 agglomerative methods and partitional methods which are used to compare effective methods, and shows that they can often be applied in vector.

Agglomerative Clustering

Agglomerative methods are a type of hierarchical clustering methods. Theses methods merge a pair of clusters at each 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. The clusterings found by agglomerative 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.

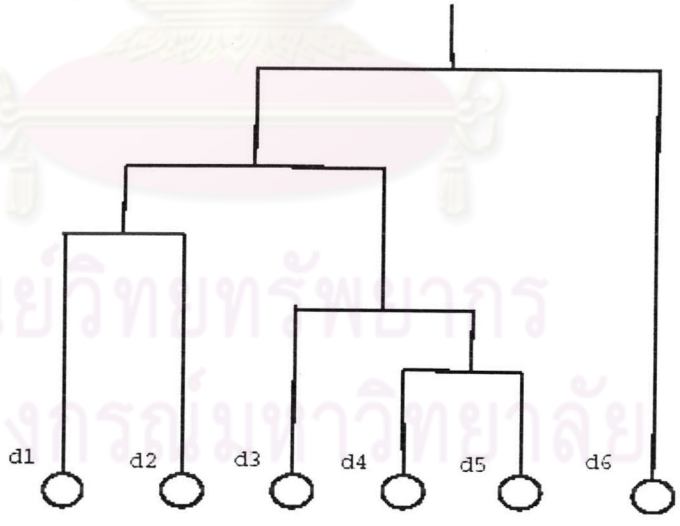

Figure 2.4: Example of Dendogram

Figure 2.4 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 2.4 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 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. The complete link criteria selects the pair with the 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. The objects with the minimum distance are considered to be the most similar.

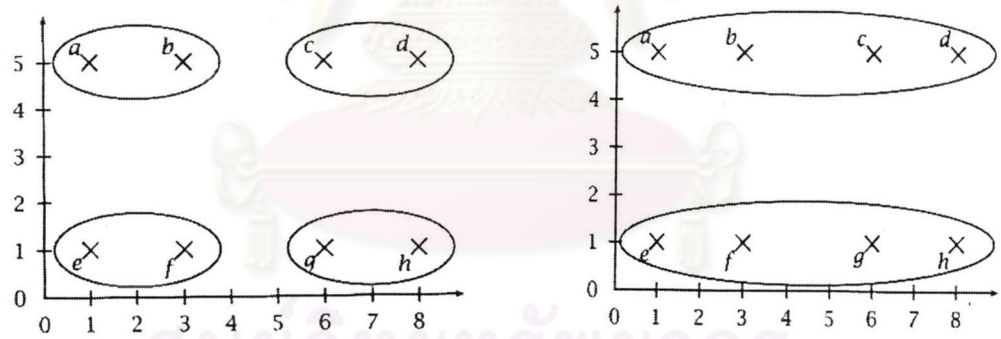

Figure 2.5: Single Link Clustering

Figure 2.5 shows how the single link criteria select the pair of clusters for merging and a cluster for splitting. In the single link diagram, the two objects that are closest to each other are used for determining the amount of similarity between the two clusters.

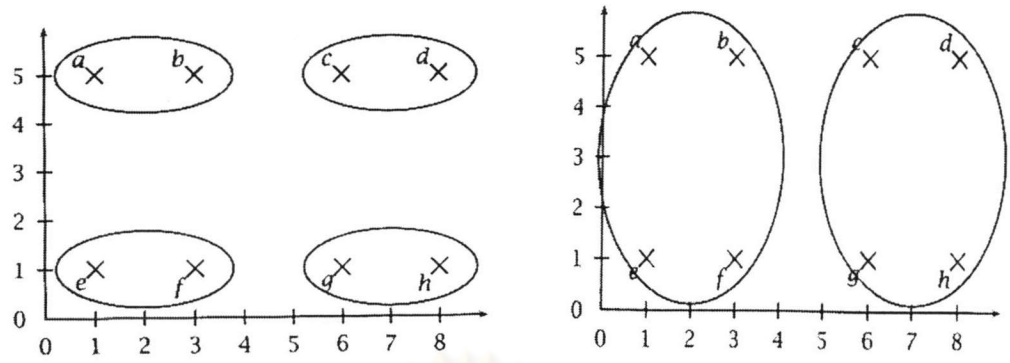

Figure 2.6: Complete Link Clustering

Figure 2.6 shows how the complete criteria select the furthest pair of objects for merging.

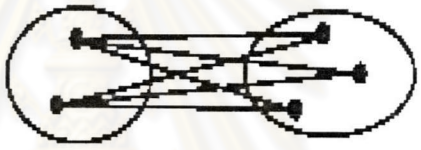

Figure 2.7: Average Link Clustering

Figure 2.7 shows the average link clustering measures the distance between the centroids of the two clusters to determine similarity.

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. The distance between a pair of vectors is determined by the angle between them, while similarity is measured by the cosine of this angle.

Partitional Clustering

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 agglomerative methods, the clusters created during subsequent iterations are not related to those in the previous or next iterations. 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. 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. The basic K-means clustering technique is presented below.

Basic K-means Algorithm for finding K clusters.

1. Select K points as the initial centroids.
2. Assign all points to the closest centroid.
3. Recompute the centroid of each cluster.
4. Repeat steps 2 and 3 until the centroids don't change.

2.4 Evaluation

For the word sense disambiguation, the evaluation of performance is measured in term of *precision*, *recall* and *F-measure*. The definitions of these terms are the same as standard measure of the information retrieval (IR) performance.

2.4.1 Precision

Precision is defined as the ratio of the number of senses correctly clustered divided by the number of answered senses which are returned by the clustering algorithm. For each word, the outputs are a list of clusters to which the sense belongs. Each cluster corresponds to a sense of the word. The precision of the system is measured by the percentage of output clusters that actually correspond to a sense of the word. To compute this precision, we must define what it means for a cluster to correspond to a correct sense of a word. The degree to which the discovered

clusters correspond with the true word senses as indicated by the sense–tags. The precision is the average precision of all senses of each word. The overall precision can be formulated as in equation 2.4.

$$\text{Precision} = \frac{\text{number of correctly clustered sense}}{\text{number of answered senses}} \qquad (2.4)$$

2.4.2 Recall

Recall is computed by dividing the number of senses correctly clustered to which word is assigned by the total actual number of senses in which word used in the data set. The overall recall is the average recall of all senses of each word. The recall can be formulated as in equation 2.5.

$$\text{Recall} = \frac{\text{number of correctly clustered sense}}{\text{total actual number of sense}} \qquad (2.5)$$

2.4.3 F-measure

The F–measure (Rijsbergen, 1979) is a standard way of combining the precision and recall values into a single figure and is in fact the harmonic mean of these two values given to both precision and recall. The F-measure can be formulated as in equation 2.6.

$$F = \frac{2 \times \text{precision} \times \text{recall}}{(\text{precision} + \text{recall})} \qquad (2.6)$$