Collocation based Word Sense Disambiguation using Clustering for Tamil

Baskaran S¹, Vaidehi V²

¹AU-KBC Research Centre, ²Department of Information Technology
Madras Institute of Technology, Anna University
Chennai, India
E-mail: baskaran@au-kbc.org, vaidehi@mitindia.edu

Abstract:
In this paper we present a unsupervised approach for Word Sense Disambiguation using clustering technique. This approach relies on the automatically extracted collocations for tagging the ambiguous words with the appropriate senses. The advantage of this approach is that the dependence of word sense disambiguation algorithms, on either a sense-tagged corpus or a knowledge-source, such as dictionary, thesaurus etc. is removed. The algorithm is first trained on a training corpus to extract relevant collocations for each ambiguous word. We use the clustering technique to group the occurrences of an ambiguous word in the training corpus, where each group corresponds to a particular sense of the ambiguous word. Prominent collocations are then identified for each group. These collocations are then tagged with the appropriate sense manually and this information is used for disambiguating unseen occurrences of the ambiguous word. This method reduces the human effort in terms of sense tagging compared to the earlier approaches and at the same time gives reasonable levels of accuracy.

1. Introduction:

Word Sense Disambiguation (WSD) is the task of assigning the appropriate sense for each occurrence of an ambiguous word in a document. WSD is important for applications such as Machine Translation (MT), Information Retrieval (IR) etc. For example, consider the sentence,

(1) vila cariyaka malai aru manikkat totangiyatu

function exactly evening six O’clock started

The function started exactly at six O’clock in the evening

In the above sentence the word malai is ambiguous and has at least two senses, viz. evening and garland. When this sentence is given for, say a Tamil-English machine translation system, it has to correctly translate this word as evening and not as garland. Similarly an intelligent IR system should be capable of distinguishing the different senses of a word in the query and should return the documents that are relevant to the intended sense of the query.
Basically there are three types of lexical ambiguities i) Polysemy, ii) Homonymy and iii) Categorial ambiguity (Hirst 1987). Polysemous words are the words having several meanings that are related to one another. For example the English verb ‘open’ has many senses concerning unfolding, expanding, revealing, making openings in and so on. Similarly in Tamil, the verb piti is polysemous having senses such as,

(2) piti : capture, catch, massage, shape / mould and smoke

It should be noted that, piti is one of the most ambiguous Tamil words. It appears in both Parts-Of-Speech (POS) categories of nouns and verbs. In the verb form, it has more than ten senses that are too subtle to easily differentiate and in the noun category it has five senses.

The second type of lexical ambiguity is called homonymy. Words having multiple meanings that are totally unrelated are called homonymous words. For example the word ‘bank’ given in the previous subsection is an example for homonymy. The Tamil word malai has two completely unrelated senses of evening and garland. Here it can be seen that the senses are clearly distinct unlike the case of polysemous words.

Categorial ambiguity forms yet another type of ambiguity. Words having multiple meanings with each meaning having different grammatical categories are called categorially ambiguous words. Consider the word atu in Tamil, which has at least two senses with each being in different POS category.

(3) atu₁ - N – goat

atu₂ - V - dance

Homonymous words are generally easy to disambiguate automatically because of their distinctness in their senses. On the contrary polysemous words are very harder to disambiguate, as their senses are too subtle to be easily differentiated from each other. Categorically ambiguous words pose another type of problem to any NLP application. While analysing these words the problem takes the new dimension of identifying the POS of the word. Thus this becomes the task for a separate POS tagger, which associates one correct POS tag with each word. In the present work the focus is to disambiguate the homonymous words in a given text in Tamil.

Earlier works on WSD used knowledge source such as a dictionary, a thesaurus or were dependent on sense-tagged corpus for disambiguation. The main problem with these approaches is that these can not be readily extended to other languages because of the non-availability of such resources. Also, because of the same reason
these techniques can not be applied to specific domains such as medicine, physics etc. Developing a lexical resource such as a dictionary or thesaurus is not only costly, but is also time consuming and painstaking. However the present approach does not require any such knowledge source, but uses the information from a corpus, which is easily available. Hence this approach can be extended for any language for disambiguation purposes.

The framework for disambiguation in this work includes the knowledge source available from the context of the ambiguous words i.e. the collocations. Collocations are the words that are adjacent to the target-word, strongly indicating the sense of the ambiguous word. This is a corpus based statistical approach to WSD and this approach circumvents the need for any lexical resource such as a dictionary, a thesaurus or a WordNet. The approach is based on clustering technique in which a number of points (occurrences of the ambiguous word in a corpus) are grouped into different clusters, in such a way that intra-cluster similarity is maximum and inter-cluster similarity is minimum or zero. The clustering process and the subsequent process of collocation extraction are performed in a completely unsupervised way. The senses are then associated manually to the different sets of collocations, to create a sense-collocation dictionary, which is then used for disambiguation.

The organisation of this paper is as follows. The section 2 outlines the earlier works on WSD. Section 3 explains the significance of using collocations and discusses the implementation details in its entirety. Evaluation of the algorithm is discussed in section 4. Conclusion is given in the last section.

2. Earlier works on WSD

Lesk (1986) was the first to start from the simple idea that words in a dictionary definition are likely to be good indicators for the senses they define. Suppose that a word has two senses, the definitions of which are available in a dictionary. Then according to him, a given new occurrence belongs to the sense whose definition matches for maximum number of common content words with that of the occurrence. Lesk reports accuracy in the range of 50% -70% for a sample of ambiguous words. Dagan and Itai (1994) proposed another algorithm based on dictionary and second language corpus. It uses a English-German bilingual dictionary and a German monolingual corpus to disambiguate English words. It identifies syntactic relations between words using a source language parser, and maps all the alternative interpretations of these relationships to the target language using a bilingual lexicon. It then counts the translations for each sense and assigns the sense for which the count is maximum. The algorithm is more complex and disambiguates only if a decision can be made reliably.
Walker (1987) proposed a simple algorithm for WSD based on thesaurus. This is based on the thesaurus categorisation where each word is assigned one or more subject codes. If a word is ambiguous then it is assigned more than one subject code. According to this approach, the words for which the thesaurus list the same subject codes are counted together, and the ambiguous word is assigned the sense (subject code) for which the count is maximum. Black (1988) reported only around 50% accuracy when he applied Walker’s (1987) algorithm. It should be however noted that in this implementation Black used test words which were difficult and highly ambiguous such as interest, point, power, state and terms (Manning and Schutze 1999). Yarowsky (1992) proposed the adaptive thesaurus-based disambiguation to improve Walker’s approach. The algorithm adds words to a particular category, if they occur more often than the chance, in the context of the category in a corpus. The method achieved high accuracy in the range of 95%-99% when thesaurus categories and senses align well with the topics. When the sense is not specific to a particular topic the algorithm could achieve only 34%. Thus this algorithm is a topic-based and works well for topically distinct words.

Ng and Lee (1996) proposed an exemplar-based learning algorithm that is based on multiple knowledge sources such as part of speech of neighbouring words, morphological form, inordered set of surrounding words, local collocations verb-object syntactic relations etc. The system called as LEXAS (LEXical Ambiguity – resolving System) performs WSD by first learning from a pre-tagged training corpus automatically extracts the feature set. Subsequently, in the test phase the system again extracts the features but from an unseen text. This feature is then compared with the feature sets obtained in the training phase. The sense of the word ‘w’ in the test example is that of the closest matching training example. Ng and Lee report the mean accuracy of the algorithm to be 87.4%. WASP-Bench, proposed by (Kilgarriff, Tugwell 2001) is a lexicographer’s workstation for developing state-of-the-art techniques for WSD aimed specifically for MT. It simplifies the lexicographers task of framing the translation rules, which will look like, “in context c, translate source language word S as target language word T”. In the WASP-bench, various parameters such as base-noun, possessive, plural, passive, subject-object, adjectival-component, pp-complement are extracted from corpus based on unary, binary and trinary relations,. The lexicographer associates the target sense for each clue in the relation by looking the actual corpus instances for reference. These associations are then used for disambiguating new occurrences.

Yarowsky (1993) used two powerful properties of one sense per collocation and one sense per discourse. The idea of one sense per collocation was that, nearby words provide strong and consistent clues to the sense of the
target word depending on relative distance order and syntactic relationship. The applicability of one sense per discourse is based on the highly reliable finding that, the sense of the target word is highly consistent within any given document. \textit{Decision list} approach by Yarowsky (1994 and 1995) was again based on these two. He reports the \textit{accuracy}\textsuperscript{1} and \textit{applicability}\textsuperscript{2} of the one sense per discourse hypothesis to be 99.8\% and 50.1\% respectively. The algorithm is bootstrapped by giving few seed words to each sense of the ambiguous words. It then extracts more collocations for each sense from a corpus. The collocations are added to the decision list, which is used to disambiguate new unseen text. The disadvantage of this approach is the manual labour involved in giving the initial seed words. The algorithm gives over 90\% accuracy for a set of twelve words used for testing.

The \textit{context group discrimination} algorithm proposed by Schutze (1998) focuses on the sense discrimination problem, which is easier than full disambiguation. The approach proposed by Schutze is completely unsupervised and is aimed for Information retrieval applications. The algorithm clusters the contents of the ambiguous word into a number of groups with each group containing contextually similar occurrences and discriminating between these groups without labeling them. An occurrence in a test text is disambiguated by computing the second-order representation of the relevant context, and later assigning it to the cluster whose centroid is closest to that representation. The unsupervised clustering based algorithm proposed by Baskaran (2002) uses the collocations and case-markers present in the corpus as knowledge sources. The algorithm first groups the occurrences of an ambiguous word into several clusters and then extract collocation and case-marker information from the clusters. These information are then used for disambiguating unseen occurrences.

3. The Approach

3.1 Significance of Collocations

Collocations are nearby words that strongly suggest the sense of the ambiguous word in a given occurrence. In general, the term \textit{collocation} refers to a quantifiable position-specific relationship between two lexical items. Collocations encode information about words that are semantically very close to the ambiguous word and hence can be reliably used for disambiguation. Consider the English word ‘plant’ having the two senses.

(4) \begin{align*}
\text{plant}_1 & \rightarrow \text{a living organism} \\
\text{plant}_2 & \rightarrow \text{a manufacturing place}
\end{align*}

\textsuperscript{1} When the word occurs more than once in a discourse, how often it takes on the majority sense for the discourse
\textsuperscript{2} How frequently the word \textit{does} occur more than once in a discourse
Collocations for the word ‘plant’ will include the words that frequently occur with ‘plant’ in texts. Some typical collocations for the above two senses of ‘plant’ are listed below.

(5) plant$_1$ – growth, height, flower, fruit, species, leaves  
    plant$_2$ – car, union, equipment, assembly, nuclear, job, worker

Similarly, consider the Tamil word $katir$ which has two senses listed below.

(6) $katir_1$ – grain  
    $katir_2$ – (light) ray

Typically collocations for $katir$ in these two different senses would be,

(7) $katir_1$ – nel (rice), kotumai (wheat), aruvatai (harvest), payir (plant),  
    $katir_2$ – curiyan (Sun), oli (light), lecar (lesar), eks (X-rays), pura uta (ultra-violet)

Generally collocations are sensitive to the distance from the ambiguous word. Words that are at a short distance from the ambiguous word strongly indicate its sense; as the distance increases the relevance of the word to that of the ambiguous word decreases. The distance is commonly referred to as window size. In the present approach we use the window size of 10-words on either side of the ambiguous word.

The basic difference between the earlier approaches and the current one is the way in which these collocations are identified. In earlier approaches collocations were induced from the dictionary or thesaurus or sometimes given manually to bootstrap the algorithm (Yarowsky 1995). In the present approach collocations are automatically extracted from the clustered occurrences. This eliminates the need to give seed collocations manually, which is a laborious task considering the large number of ambiguous words in a language.

3.2 System Architecture

The working of the system is shown in its entirety in the system flow diagram in Figure 1. The separate stages of the system are explained in the next chapter with the details of implementation. The approach works in two phases, viz. training phase and testing phase. In the training phase the algorithm is trained using the clustering technique to group the occurrences of the ambiguous words. It later extracts the collocations for each cluster automatically. The collocations are now associated with the appropriate senses by human annotators to form the sense-collocation dictionary. This dictionary is then used in testing phase to resolve the ambiguities in an unseen document.
3.3 Context, Context Space and Context Vector

*Context* can be defined as the content words falling within a specific window centered on the ambiguous word. Usually the window size varies depending on the application of the underlying statistical technique. For each
occurrence of an ambiguous word a context is identified from the text. Different researchers have used different sizes of contexts, i.e. different window sizes. The window size influences the performance of the system considerably. A larger window size brings in more unrelated words to the context, while a smaller window size misses some important collocations. Following the general practice as found in many WSD works including Yarowsky (1995), a twenty-word window is used in this work.

All the contexts of an ambiguous word in a large corpus can be represented in a Context Space denoted by the notation ‘S’. In other words, the context space $S$ is defined to consist of a set of points, with each point denoting a unique context of the ambiguous word. Figure 2. illustrates the representation of the context space.

All the contexts are represented as context vectors, which correspond to individual points in the context space. In forming the context vectors, content words$^3$ alone are considered leaving out the function words$^4$. All the words that fall within the context window are morphologically analysed and the resulting root words are further pruned to remove the stop words and the remaining words are included in the list of content words. This is repeated for all the occurrences so as to identify all the content words. Now context vectors are generated for each occurrence by substituting for each content word, either 1 or 0, depending on whether it is present in that particular occurrence or not.

Figure 2 – Representation of Context Space S

---

$^3$ Words that occur not too frequently and that have more relevance to a particular domain are called the content words

$^4$ Words that occur more frequently in any document irrespective of the domain. In English a, the, in, on, and etc. are some of the functional words. They are also called as stop words.
3.4 Clustering

Clustering process is executed in two phases. In the first phase Group Average Agglomerative Clustering (GAAC) is employed to a reduced number of occurrences (i.e. context vectors) to obtain k-initial clusters, where, k is the number of senses of the ambiguous word decided by the user or inferred from a dictionary. These k-initial clusters are then used as seeds to K-means clustering in the second phase that produces k-final clusters. These are then analysed separately to extract collocations from each of these clusters. Prominent collocations from each set are then given to human annotators who tag the appropriate sense to each collocation set, to form the sense-collocation dictionary.

3.4.1 Group Average Agglomerative Clustering (GAAC)

Agglomerative clustering is a hard clustering algorithm, which starts with separate cluster for each object, i.e. the context vector, to be clustered and at each step merges two closest objects to form a single cluster. This is computationally much complex in terms of time taken for clustering. The time complexity for this approach is $O(n^2)$, to cluster ‘n’ objects. This will create difficulty due to the large value of ‘n’, typically around one thousand. Because of its higher time complexity, this is applied for reduced number of context vectors and the resulting k-clusters are used as seeds to the K-means clustering.

First ‘n’ context vectors are chosen at random out of the total ‘$n^2$’ context vectors available. At the beginning these context vectors are considered as independent clusters and distance between each of them is calculated. Two clusters that are of least distance apart are merged. This is repeated till we are left with a pre-determined number of clusters that is equal to the total number of senses of the ambiguous word. The distance between any two clusters is calculated as the average distance between them using the formula, i.e. as the distance between the cluster centers. Cosine is used as the distance measure, whereby, the distance between two vectors $V_i$ and $W_i$ is measured as the cosine of the angle between two cluster centers, as below.

$$\cos(V,W) = \frac{\sum_{i=1}^{N} V_i \cdot W_i}{\sqrt{\sum_{i=1}^{N} V_i^2 \cdot \sum_{i=1}^{N} W_i^2}}$$

The cluster center is determined by,
\begin{equation}
\mu = \left( \frac{1}{|C_j|} \right) \sum_{x \in C_j} x
\end{equation}

The stopping criterion for the algorithm is the reaching point of desired number of clusters.

### 3.4.2 K-means clustering

K-means clustering begins with k-initial cluster centers (seed cluster centers) and is executed iteratively, and in each iteration two steps viz. i) assignment step and ii) recomputation step, are involved. In the assignment step the algorithm assigns each of the cluster to a closest cluster center. After assigning all the clusters in this way, cluster centers are recomputed for the new assignment by taking the average over all the points in the cluster, which forms the recomputation step. These new average points are now taken as representing the cluster for the next iteration. This is repeated iteratively till there is no reallocation of cluster points in any two subsequent iterations. Here again, the two formulae given in equations 3.7 and 3.8 are used to measure the closeness between two clusters and to calculate the cluster center respectively.

### 3.5 Extracting Collocations

At the end of clustering, groups of cluster are obtained with each being contextually similar. The task now is to identify the potential collocations, which will indicate the sense of the ambiguous word reliably. Also these words should be ordered to give high priority to the most vital and appropriate collocation, the presence of which clearly identifies the correct sense without even a small degree of doubt. Similarly certain collocations, which still indicate the sense reliably but occur rather very rarely, should be given a relatively low priority. The collocations are ranked based on the log-likelihood ratio (Yarowsky 1995) given by,

\begin{equation}
\text{log} \left( \frac{P(Sense_A | Collocation_i)}{P(Sense_B | Collocation_i)} \right)
\end{equation}

This equation measures the importance of a collocation to a particular sense relative to its importance for the other senses. Thus a collocation that is equally probable in all the clusters and thus in all the senses, will be given near zero priority. A threshold value is used for the log-likelihood ratio, to determine whether a word can be used as a collocation or not. Thus, words having the log-likelihood value higher than the threshold alone are considered as collocations. Now, consider the word *vell* and two of its major senses, viz. *Friday* and *Venus*. 

---

10
For these two senses, the collocations such as cevvay (Tuesday), butan (Wednesday), viyalan (Thursday) and cani (Saturday) occur commonly and hence these cannot used reliably to identify the sense. These collocations have equal probability of occurrence for both senses and thus they fail to distinguish between these senses. Thus the numerator and denominator of the log-likelihood measure given in 3.9, become comparable to each other resulting in a value close to ‘1’ and the log-likelihood value approaches ‘0’. This suggests that these words cannot be used reliably as collocations as the use of these words is likely to mislead the algorithm thereby reducing the performance of the system.

Now, consider another example, malai having two senses evening and garland. kalai is an important collocation for the ambiguous word for the evening sense. kalai will occur more frequently in one particular cluster but only once or twice in the other cluster. The log-likelihood measure will now have a high value because of the high relevance of the word to one cluster. Thus the word kalai turns out to be an important collocation for the word malai.

The list of collocations extracted by the algorithm for the two clusters obtained for malai are shown below in Figure 2. The list is a partial list showing only top collocations along with their meanings for each sense.

(11) Cluster #1: kalai (morning), pakal (daytime), curiyan (Sun), marai (Sun) set
(12) Cluster #2: pu (flower), manam (fragrance), roja (rose), mallikai (jasmine)

3.6 Associating Senses with Collocations

Having extracted the collocations from the clusters, the next step is the task of sense tagging. Here the human annotators are given the top ten collocations that were extracted from the clusters and also the senses of the ambiguous word. The words are split into different groups and each group is given to different annotators. Totally eight annotators are given the task of sense tagging. The task of the annotators is to build a sense-collocation dictionary by associating the more appropriate sense to each collocation set, by looking the top collocations from each cluster i.e. collocation set. In the rare case of inability on the part of the annotator in identifying the senses, more collocations are given to assist the annotator. Sample entries in the sense-collocation dictionary are shown in Figure 3. The fields in the sense-collocation dictionary correspond to i) the ambiguous word, ii) sense assigned manually by the annotators (based on the collocations) and iii) collocations extracted by the algorithm. The senses associated by the annotator are underlined for the purpose of illustration. Also shown for illustration are the English gloss between parentheses and the roman transliteration in italics.
4. Evaluation of the system

The complete algorithm was trained and tested in a 3 million word Tamil corpus, provided by the Central Institute of Indian Languages (CIIL), Mysore. We divided the corpus into two parts in ratio of 80 to 20. Following the standard procedure we trained the algorithm on 80% of the corpus and used the remaining 20% for testing purposes. In the testing phase the sense-collocation dictionary is used for disambiguating unseen occurrences. The collocations that are found in the training phase are used to resolve the ambiguities in the given text. However it should be noted that the algorithm should have been previously trained for the ambiguous words that appear in the given text.

The given text is first analysed morphologically to cut off the inflections and to get the root forms. The system then identifies the words that are ambiguous by using the dictionary built in the training phase. The task now is to tag the occurrences of the ambiguous word with the sense given in the sense-collocation dictionary. To do this contextual words in the given text are compared with the collocation words listed for the ambiguous word.

Figure 3 – Sample entries in the sense-collocation dictionary
in the dictionary. When a context word is found to be present in the dictionary, its corresponding sense will be assigned to the particular occurrence of the ambiguous word.

The algorithm has been tested on eight ambiguous words; the results are shown in Table 1. For each ambiguous word only two major senses are considered. The first and second columns in the table show the ambiguous word and two major senses. The sample size, which indicates the total number of occurrences of the ambiguous word available in the 80% of the corpus used for training the algorithm is given in the third column. The fourth column shows the *baseline* for each word. The fifth column titled ‘accuracy’ displays the performance of the algorithm in disambiguating the unseen raw occurrences from the 20% test corpus.

**Table 1 - Performance of WSD system using collocations**

<table>
<thead>
<tr>
<th>Word</th>
<th>Senses</th>
<th>Sample size</th>
<th>Major sense %</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>malai</td>
<td>evening / garland</td>
<td>642</td>
<td>72.1</td>
<td>84.0</td>
</tr>
<tr>
<td>vilangu</td>
<td>animal / hand-cuff</td>
<td>378</td>
<td>69.3</td>
<td>76.9</td>
</tr>
<tr>
<td>kadir</td>
<td>rays / grain</td>
<td>779</td>
<td>60.8</td>
<td>80.4</td>
</tr>
<tr>
<td>nul</td>
<td>book / thread</td>
<td>3158</td>
<td>72.7</td>
<td>84.1</td>
</tr>
<tr>
<td>tanti</td>
<td>telegram / string</td>
<td>272</td>
<td>51.1</td>
<td>75.6</td>
</tr>
<tr>
<td>varicai</td>
<td>queue / offerings (marriage)</td>
<td>521</td>
<td>53.8</td>
<td>80.6</td>
</tr>
<tr>
<td>katai</td>
<td>story / ancient weapon</td>
<td>2265</td>
<td>73.6</td>
<td>87.2</td>
</tr>
<tr>
<td>tuntu</td>
<td>towel / piece</td>
<td>337</td>
<td>63.4</td>
<td>78.3</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>1044</td>
<td>64.6</td>
<td>80.9</td>
</tr>
</tbody>
</table>

It can be observed from the table that the algorithm works well for the words such as *malai*, *nul* and *katai* giving nearly 84% accuracy. But words such as *tanti*, *tuntu* and *vilangu* performed poorly. This is probably due to the data sparseness in the corpus for these words. The other two words *kadir* and *varicai* performed moderately with nearly 80% accuracy. It can be observed from these results that the performance of the WSD system improves considerably with the increase in the size of the corpus.

An experiment was performed to compare the manual labour involved in decision tree approach (Yarowsky 1995) and the present approach. For this experiment ten ambiguous Tamil words are chosen; and these ten words are split into three groups and each group was given to different experts. The experts were also provided...
the senses of these ambiguous words and were asked to come up with one or two collocations for each sense as in (Yarowsky 1995). To avoid any bias in inferring the collocation word from the sense, the senses were provided in English.

Table 2 - Comparison of the present approach with manual approach - in selecting seed words

<table>
<thead>
<tr>
<th>Word</th>
<th>Majority Senses</th>
<th>Manually assigned Seed Words Tamil (English)</th>
<th>Seed words found in top 20 collocations extracted automatically</th>
</tr>
</thead>
<tbody>
<tr>
<td>varnam</td>
<td>Colour / paint</td>
<td>cuvar (wall), oviyam (painting)</td>
<td>oviyam, cuvar</td>
</tr>
<tr>
<td></td>
<td>Type of song</td>
<td>rangam (tune)</td>
<td>rangam</td>
</tr>
<tr>
<td>malai</td>
<td>garland</td>
<td>pu (flower), malar (flower)</td>
<td>pu, malar</td>
</tr>
<tr>
<td></td>
<td>evening</td>
<td>mani (time), nerru (yesterday)</td>
<td>mani, nerru</td>
</tr>
<tr>
<td>karai</td>
<td>River bank</td>
<td>aru (river), min (fish)</td>
<td>aru, min</td>
</tr>
<tr>
<td></td>
<td>Border of a cloth</td>
<td>tuntu (towel), akalam (broad)</td>
<td>tuntu</td>
</tr>
<tr>
<td>velli</td>
<td>Metal / chemical</td>
<td>tangam (gold), nagai (jewellery)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>planet</td>
<td>girakam (planet), curiyan (sun)</td>
<td>curiyan</td>
</tr>
<tr>
<td></td>
<td>Friday</td>
<td>kilamai (day), varam (week)</td>
<td>kilamai</td>
</tr>
<tr>
<td></td>
<td>medal</td>
<td>olimpik (olympic), potti (competition)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>silver jubilee</td>
<td>tirumanam (marriage)</td>
<td></td>
</tr>
<tr>
<td>vilangu</td>
<td>animal</td>
<td>katu (forest), cingam (lion)</td>
<td>katu, cingam</td>
</tr>
<tr>
<td></td>
<td>hand-cuff</td>
<td>polis (police)</td>
<td>polis</td>
</tr>
<tr>
<td>nul</td>
<td>book</td>
<td>aciriyar (author)</td>
<td>aciriyar</td>
</tr>
<tr>
<td></td>
<td>thread</td>
<td>tuni (cloth), ney (weave)</td>
<td></td>
</tr>
<tr>
<td>katur</td>
<td>grains</td>
<td>nel (rice), murri (grown)</td>
<td>nel</td>
</tr>
<tr>
<td></td>
<td>rays (of Sun)</td>
<td>oli (light), curiyan (sun)</td>
<td>oli, curiyan</td>
</tr>
<tr>
<td>tanti</td>
<td>string (instrument)</td>
<td>vayalin (violin), vinai (veena)</td>
<td>vayalin, vinai</td>
</tr>
<tr>
<td></td>
<td>telegram</td>
<td>avacaram (urgent), ceyti (news)</td>
<td>ceyti</td>
</tr>
<tr>
<td>kuttai</td>
<td>short</td>
<td>paiyan (boy), uruvam (appearance)</td>
<td>uruvam</td>
</tr>
<tr>
<td></td>
<td>pond</td>
<td>tannir (water)</td>
<td>tannir</td>
</tr>
<tr>
<td>kampu</td>
<td>stick</td>
<td>kilavar (old man)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>millet rice</td>
<td>kelvaraku (ragi), taniyam (grain)</td>
<td>kelvaraku</td>
</tr>
</tbody>
</table>

Separately, the collocations extracted by the clustering process along with the senses are given to the experts and now, they were asked to assign the appropriate sense to each set of collocations. In most of the cases the clustering algorithm automatically extracted the seed words given by the experts. This suggests that the manual
labour in assigning seeds can be replaced by the clustering algorithm as being used here. The results of the experiment are shown in the Table 2 above.

The first column contains the ambiguous words and the second column, their senses. The third column contains the seed words given by the experts, along with their English meaning (in brackets) and finally the fourth column contains the seed words found in top twenty collocations that were automatically extracted by the algorithm. The table clearly indicates that the seed words given by the experts were also automatically extracted by the algorithm in most of the cases. The algorithm fails in cases where the senses are too subtle to be separated from the other occurrences of the ambiguous word. For example for the word *velli*, the senses ‘metal’ and ‘medal’ are very close. The algorithm also fails when the training data is sparse, as for one sense in words such as *velli*, *nul* and *kampa*.

5. Conclusion

The use of clustering for Word Sense Disambiguation is relatively a new idea and it has been implemented successfully here for disambiguating homonymous words in a given Tamil text. Apart from the use of the clustering algorithm, this approach makes use of collocations as a knowledge source. The approach can be viewed, on the one hand- as using statistical techniques for the clustering process and; on the other hand- as using information extracted automatically from a raw corpus, in the form of collocations, for disambiguation.

The complete algorithm has been implemented and tested with a 3-million word Tamil corpus. The results obtained are quite encouraging. The highest accuracy of the algorithm is around 87% (Table 1) and the average accuracy of the algorithm is around 81% (Table 1), which fare well with the other knowledge-based approaches to WSD (Lesk, 1986; Walker 1987; Black 1988; Dagan and Itai 1994). The accuracy can be improved to near 90% with further research and improvements such as 1) using a bigger corpus, and 2) by applying the *one sense per discourse* hypothesis (Yarowsky 1995) for the unresolved occurrences.

This approach demonstrates a substantial reduction in the labour of human-annotators compared to Yarowsky (1995) for a WSD system. This is achieved mainly because of the clustering algorithm used here to identify the collocations automatically. Thus, this method eliminates the need for giving seed words manually, which is a laborious process.
Acknowledgements

We would like to thank Dr. S Rajendran, Tamil University, Dr. S V Ramanan, AU-KBC Research Centre and Dr. Vijay K. Shanker, University of Delaware for their guidance. We would also like to thank the members of the language-processing group in AU-KBC Research Centre, especially Dr. Sobha L. Nair for their support.

References


Key Words: Word Sense Disambiguation, Clustering, Collocations