CHAPTER V

DESIGN OF QUERY TRANSLATION SYSTEM

Query Translation (QT) for CLTR has become one of the most widely used techniques for accessing documents in different languages using a query. The combination of query translation algorithms with monolingual IR systems is a promising field to improve the performance of CLTR systems. The performance of CLTR system depends heavily on the performance of QT. While considering Indian languages, many CLTR experiments have been carried out using word-by-word translation approach and only few experiments have done in specific domains using other approaches (Thenmozhi and Aravindan, 2009). This approach faces challenges during the handling of tasks like ambiguity, named entity, Part-of-Speech tagging with Tamil-English translation (Rao and Devi, 2010; Saravanan *et al*., 2011). Phase II of the research work aims to propose algorithms that addresses these challenges in order to improve the QT process.

Several approaches for QT have been proposed, which can be grouped into three major groups. They are, machine translation approaches (Okpor, 2014), empirical (corpus or knowledge) based approaches like example based approaches and statistical based approaches (Maria Gabriela, 2005) and hybrid approaches (www.euromatrix.net/deliverables/Euromatrix_D1.3_Revised.pdf). This research work proposes the use of hybrid approaches that integrates more than one approach to take advantage of their merits and compensate their weaknesses. The proposed hybrid approach combines rule-based machine translation approach and statistical approach to perform Tamil-English Query Translation. This system, referred to as Tamil Query Translation System (TQTS) in this research, consists of several key tasks that are to be performed in a sequential order to effectively convert the Tamil query to its English equivalent. These tasks are listed below and the architectural flow is presented in Figure 5.1.

(i) Tokenization
(ii) Pre-processing
(iii) Translation
(iv) Transliteration and error correction

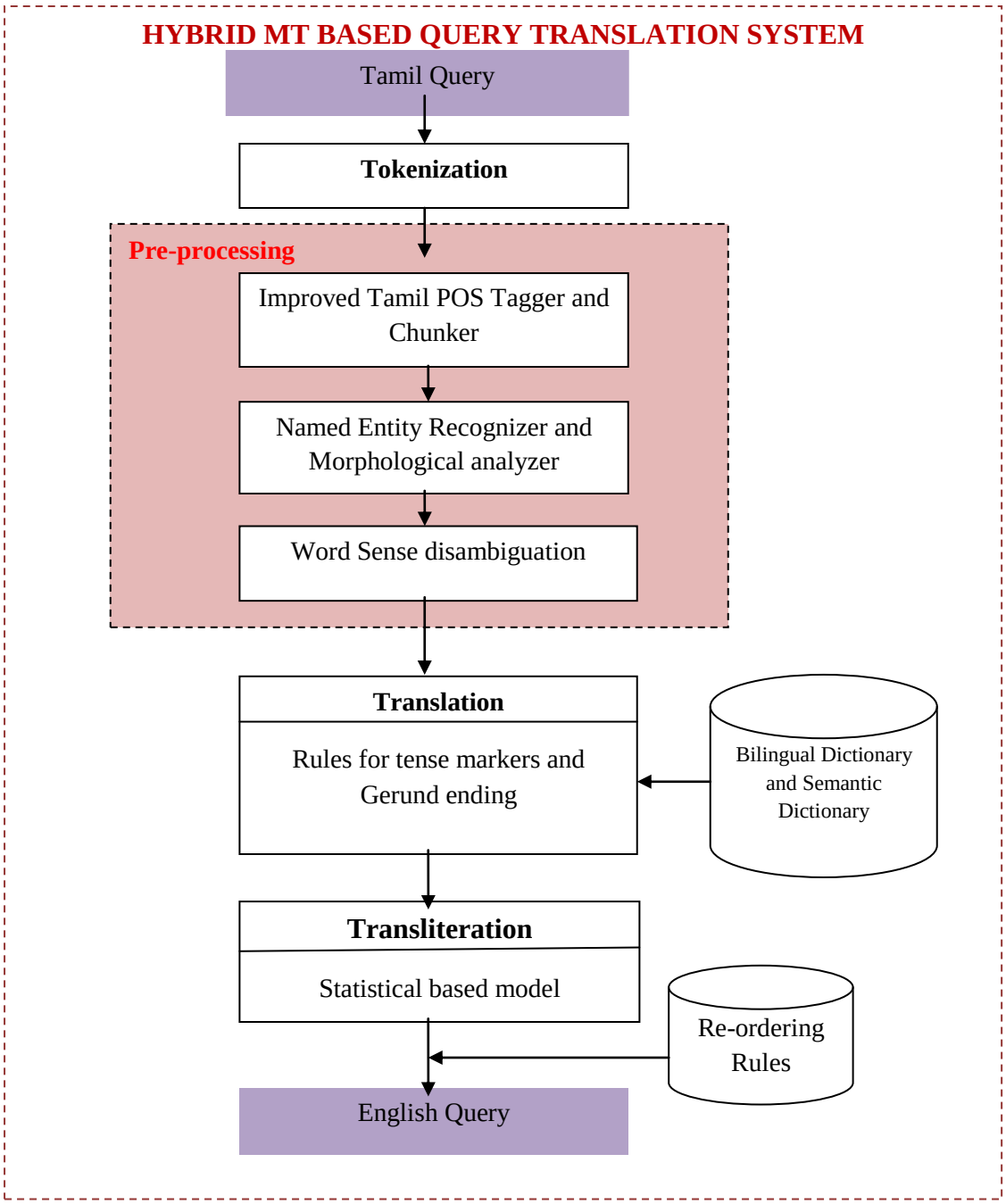

Figure 5.1: Architecture of TQTS

5.1. TOKENIZATION

The first step of TQTS is tokenization, which is the process of breaking up the query text into units called tokens (words). This process generally use some special symbols like

punctuation marks (eg. ; or -) or spaces as delimiters during word separation. This research work uses blank space as word separator. The procedure used is in Figure 5.2.

```
Let S be the recognized text
Token []= {}; wordcount = 0;
for i= 1 to Length(S)
      if char(i)  != ‘ ‘
         then tempword = tempword & char(i)
         else Token(wordcount) = tempword
      endif
endfor
```

Figure 5.2: Tokenization Pseudo-code

5.2. PRE-PROCESSING

The pre-processing step of TQTS performs five major tasks, namely, Tamil POS Tagging, Chunking, Named Entity Recognition, Morphological Analysis, and Word Sense Disambiguation. This section presents the proposed algorithmic details of these tasks.

5.2.1. POS Tagging

The first step in pre-processing of any language sentence is to retrieve Part Of Speech information that helps in processing many language related activities (Dhanalakshmi *et al*., 2009). POS tagging is defined as a task that reads a set of texts and assigns part of speech label to each of them. An example of a Tamil sentence along with the POS tagged information is given below.

Tamil Sentence : அவன் அலுவலகத்தை நோக்கி நடந்தான்

Pos Tagger : <NNP> <CL> <IN> <VM>

Approaches for performing POS tagging include rule-based, statistical-based, transformation-based and machine learning-based. Out of these, the POS tagging approach based on machine learning algorithms have attained high accuracy in English and European languages

(Hasan *et al*., 2007; Selvam and Natarajan, 2009). Motivated by these results, this research work analyzes the use of ensemble classification models for POS tagging. The proposed system termed as POS Tagging using Hybrid SVM-WNN Ensemble Classifier with Ensemble Feature Selection (POSHES). The proposed POSHES uses the standard POS tagset designed by IIIT, Hyderabad, India (Table 5.1).

TABLE 5.1

POS TAGSET

S.No	TAG DESCRIPTION	TAG	S.No	TAG DESCRIPTION	TAG
1	Noun	<NN>	14	Quantifier	<QF>
2	Noun Location	<NST>	15	Cardinal	<QC>
3	Proper Noun	<NNP>	16	Ordinal	<QO>
4	Pronoun	<PRP>	17	Common Noun	<CL>
5	Demonstrative	<DEM>	18	Intensifier	<INTF>
6	Verb Finite	<VM>	19	Interjection	<INJ>
7	Verb Aux	<VAUX>	20	Negation	<NEG>
8	Adjective	<JJ>	21	Quotative	<UT>
9	Adverb	<RB>	22	Symbol	<SYM>
10	Postposition	<PSP>	23	Compounds	<C>
11	Particles	<RP>	24	Reduplicative	<RDP>
12	Conjunction	<CC>	25	Echo	<ECH>
13	Question Words	<VQ>	26	Unknown	<UNK>

The proposed machine learning approach for POS tagging consists of three major steps.

- Feature extraction
- Feature Selection

- Classification

The following section presents details regarding these three steps.

- **Feature Extraction**

The first feature extracted is the Context Words Features of a Token (CWFT). It is defined as the preceding and following words of a particular token. This feature provides useful information during output label decision of the current token. The CWFT of the i^{th} feature is represented in Equation 5.1.

$$CWFT_i = w_{i-m}, ..., w_{i-1}, w_i, w_{i+1}, w_{i+n} \quad (5.1)$$

where w_{i-m} and w_{i+n} are the previous m^{th} and n^{th} next word of token i.

The second and third features extracted are the prefix and suffix of the token under consideration. These two feature set consists of n-length prefixes or suffixes surrounding a token. For example, prefixes of length upto 3 characters of the word '**அலுவலகத்தை**' are '**அ**', '**அலு**' and '**அலுவ**'. This feature set can be empty (or not defined) under three conditions as listed below, that is when the token under consideration has

- has length less than the length, n
- is a punctuation symbol
- has special symbols or digits

The fourth set of features extracted is the POS information regarding the previous token. The fifth feature set consists of five types of lexicon features defined as below.

- If the current word is found to appear in the lexicon with the 'noun' PoS, then the feature 'Lexicon' is set to 1.
- If the current word is found to appear in the lexicon with the 'verb' PoS, then the feature 'Lexicon' is set to 2.
- If the current word is found to appear in the lexicon with the 'adjective' PoS, then the feature 'Lexicon' is set to 3.

- If the current word is found to appear in the lexicon with the ‘pronoun’ PoS, then the feature ‘Lexicon’ is set to 4.
- If the current word is found to appear in the lexicon with the ‘indeclinable’ PoS, then the feature ‘Lexicon’ is set to 5.

The fifth feature extracted is the digit feature. If a token consists of all digits, then this feature is set to one else it is set to zero. The next feature extracted is symbol, which is set to one if the token has special characters, else it is set to zero. The seventh feature is the length feature, which is used to distinguish proper nouns from the other words. Analysis revealed that short words are rarely proper noun. In this research, if the length of the token is less than three, then it is set to 1 else it is set to zero.

The frequency feature uses a frequency list consisting of token that of frequently occur (occurring more than 10 times) in the training data. The frequency feature is set to one if the token is in the frequency list, else it is set to zero. The function feature also uses a list which consists of function words consisting of words of indeclinable classes, i.e. of preposition, interjections and conjunctions This feature is set to one if the token is in the list else it is set to zero. The last feature consist of the Chunk information, that is, Chunk tag(s) of the current and/or surrounding token(s) while considering a particular token. Thus, the first step of POSHES extracts 11 types of features as summarized in Table 5.2.

- **Feature Selection**

Feature selection is a term commonly used to describe algorithms that reduce feature vector to a manageable size for further processing and analysis. It is an active research area in pattern recognition, statistics and data mining communities, where the main idea is to choose a subset of input variables by eliminating features with little or no predictive information. Correct usage of feature selection algorithm can significantly improve the accuracy and provide a fast classification model for text recognition. For this purpose, three frequently used feature selection algorithms are used. They are Decision Tree, F-Score and Linear Discriminate Analysis.

TABLE 5.2

DESCRIPTION OF FEATURES FOR TAMIL POS TAGGING

Feature	Description
Context words	$CWFT_i = w_{i-m}, ..., w_{i-1}, w_i, w_{i+1}, w_{i+n}$
Prefixes	Prefix string of length n of word w_i
Suffixes	Suffix string of length n of word w_i
POS features	POS (w) = POS tags of previous words w_{i-1}
Lexicalized features	$= \begin{cases} 1 & \text{if wi Lexicon with NN POS} \\ 2 & \text{if wi Lexicon with VV POS} \\ 3 & \text{if wi Lexicon with ADJ POS} \\ 4 & \text{if wi Lexicon with PR POS} \\ 5 & \text{if wi Lexicon with UN POS} \end{cases}$
Digit feature	$= \begin{cases} 1 & \text{if Token has digit} \\ 0 & \text{otherwise} \end{cases}$
Symbol feature	$= \begin{cases} 1 & \text{if Token has special character} \\ 0 & \text{otherwise} \end{cases}$
Length feature	$= \begin{cases} 1 & \text{if length of Token} \geq 3 \\ 0 & \text{otherwise} \end{cases}$
Frequency	$= \begin{cases} 1 & \text{if } \bar{w_i} \text{ in frequent word list} \\ 0 & \text{otherwise} \end{cases}$
Function word list	$= \begin{cases} 1 & \text{if } \bar{w_i} \text{ in function word list} \\ 0 & \text{otherwise} \end{cases}$
Chunk info	Chunk information of current word, w_i

The F-score feature selection algorithm is a simple but effective technique that determines discriminative value for each feature in feature subset and a feature with higher F-Score value has more discriminative power. Given training samples X_k, k=1,2….m, the number of positive and negative instances is given by n^+ and n^- respectively then the i^{th} feature vector of F-Score is calculated using Equation (5.1).

$$F(i) = \frac{\left(\overline{X_i^{(+)}} - \overline{X_i}\right)^2 + \left(\overline{X_i^{-(-)}} - \overline{X_i}\right)^2}{\frac{1}{n_+ - 1}\sum_{k=1}^{n_+}\left(X_{k,j}^{(+)} - X_i^{-(+)}\right)^2 + \frac{1}{n_- - 1}\sum_{k=1}^{n_-}\left(X_{k,i}^{(-)} - X_i^{-(-)}\right)^2} \qquad (5.1)$$

where $\overline{x_i^{(+)}}$, $\overline{x_i^{(-)}}$, $\overline{x_i}$, $x_{k,i}^{(+)}$, $x_{k,i}^{(-)}$ represents positive, negative, whole sample, i^{th} feature vector of k^{th} positive and negative instances respectively. The numerator indicates discrimination between positive and negative sets, and denominator is sum of deviation within each of feature sets (Chen and Lin, 2006). The larger the F-score value is, the more likely this feature is more discriminative.

The second feature selection algorithm is the Linear Discriminant Analysis, which was first introduced by Ronald A. Fisher and therefore, is also called as Fisher Linear discriminant. The goal of LDA is to perform dimensionality reduction without loss of any information. The LDA constructs one or more discriminate equations D_i from linear combination of predictor variables X_k such that different groups differ as much as possible as D. The LDA is expressed in Equation (5.2).

$$D_i = b_0 + \sum_{k-1}^{p} b_k x_k \qquad (5.2)$$

where D represents discriminate score. The Linear Discriminant Analysis (LDA) finds the feature that best separates all the classes using a discriminate score.

The decision tree approach for optimal feature selection was introduced by Quinlan and their variables values representation is in the form of tree, which consists of root, branches, and leaves. The first step in construction of root node is to select a best test attribute done through a measure called information gain, calculated for all attributes, and the node having highest information gain is considered as root node. The splitting criterion is applied from root node to

produce branches or subsets, and this splitting procedure is repetitively iterated to divide the subsets further. This procedure will continue until all tuples in subset belong to same class or if split reaches to a possible extent. The bottom nodes in the decision tree are called leaves, then for each leaf decision rule will provide a unique path to reach a class label. The decision tree induction algorithm is greedy in nature that constructs tree in top-down fashion recursively. The criteria to select best test attribute for each node is using Information Gain which finds expected reduction in entropy or uncertainty, and also Entropy is a common way to measure the impurity. Here higher Entropy gives more the information content and it is presented in Equation (5.3).

$$(S) = \text{Entropy} \sum_i - p_i \log 2\ p_i \qquad (5.3)$$

where p_i is the probability of class i, in Equation (5.3).

$$Information Gain(S, A) = E(S) - \sum_{values(A)} \frac{|S_V|}{|S|} E(S_V) \qquad (5.4)$$

In Equation (5.4), E is an entropy, Values(A) represents the set of all possible values of attribute A, and S_v the subset of S for which attribute A has value v, S_v = {s in S | A(s) = v}. To construct the decision tree, best attribute is selected from all the attributes based on heuristic algorithm with maximum information gain. Suppose D consists of 50, 30, 10 and 10 samples belong to noun, verb, adjective, and adverb respectively. The original entropy of above samples is defined in below implementation.

$$H[D] = -\frac{50}{100}\log_2\frac{50}{100} - \frac{30}{100}\log_2\frac{30}{100} - \frac{10}{100}\log_2\frac{10}{100}\log_2 - \frac{10}{100}\log_2\frac{10}{100} \qquad (5.5)$$

$$\left.\begin{array}{l} Gain(D, context) \equiv H[D] - H_{context}[D \\ Gain(D, prefix) \equiv H[D] - H_{prefix}[D] \\ . \\ . \\ . \\ Gain(D, sen\,inf\,o) = H[D] - H_{sent\,inf\,o}[D] \end{array}\right\} Max(Gain) \qquad (5.6)$$

For attribute A_i with v feature values, the root of the current tree will partition D into v subsets D_1, D2…..D_v. Then the expected entropy for partitioning each A_i is determined (i.e. $H_{context}$, H_{prefix}, H_{suffix}, $H_{POSfeature}$, $H_{lexfeature}$, H_{Digit}, H_{Symbol},H_{length}, $H_{frequency,}$ $H_{function,}$ $H_{seninfo}$). Then

the attribute with highest information gain is selected as branch or root or split tree. A decision tree can be converted to a set rules and finding the best tree is using NP-Hard. The partitioning stops when there is no remaining attributes for further split.

- **Ensemble Feature Selection**

The proposed ensemble approach combines the above three feature selection approaches, which takes the whole training data as input and produces as output three reduced optimal features. These features are fused to form single feature vector using a simple union operation. This feature vector is then used to train and test the proposed SVM-WNN ensemble classifier. The process of the proposed ensemble feature selection algorithm is presented in Figure 5.3.

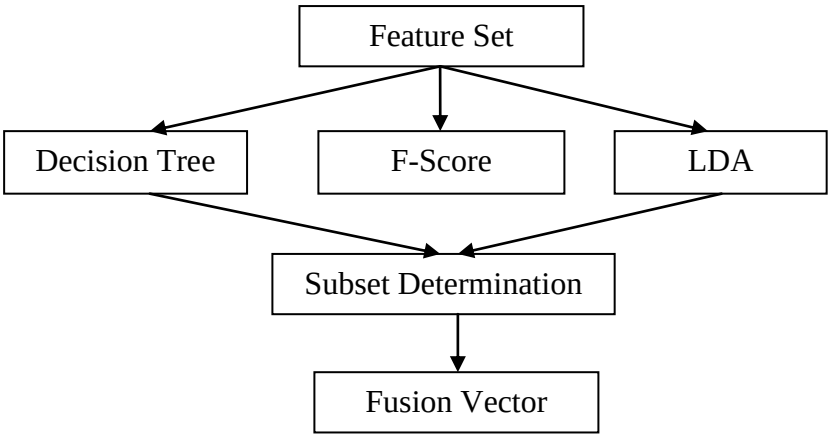

Figure 5.3: Ensemble Feature Selection

- **Ensemble Classification**

In this step, an ensemble classifier build using a hybrid SVM-WNN base classifier is used. Through analysis of related works in Tamil POS tagging the Support Vector Machine can be considered as an efficient tool which provides greater maximum accuracy, but it takes lot of time in processing the data. The Wavelet Neural Network is a powerful non-linear tool that provides better approximation facility than multilayer perceptron and radial basis function networks (Huang and Cui Baotong, 2005).

- **Wavelet Neural Networks**

Wavelet Neural Network is a classifier that has been used in several applications. The popularity is due to its great improvement over the weaknesses of neural networks. The usage of WNN has brought in the advantage of merging the structure of neural network into wavelets which extends the ability to approximate complicated patterns. The WNN can be considered an expanded perceptron in which the neurons of the first layer are replaced by wavelet nodes. The wavelet nodes allow the detection of the transient, as well as the extraction and selection of a small number of meaningful features; the obtained features are then regarded as inputs to the subsequent neurons. The WNN is designed as a three layer structure consisting of an input layer with n inputs, hidden layer with k wavelets and output layer with K outputs (Figure 5.4).

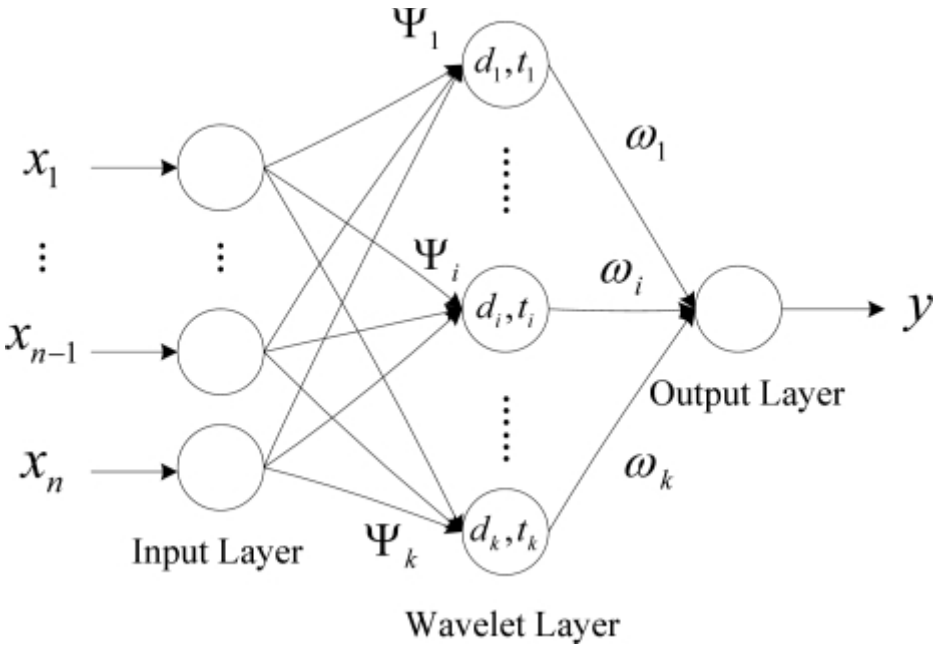

Figure 5.4 : Architecture of WNN

The architecture of wavelet network is proximate with that of multi-layer perceptrons. The fundamental difference between the multi-layer perceptrons and the wavelet network is the nature of the activation functions (sigmoidal function and wavelet basis function respectively) used by the hidden neurons. Both input layer and output layer are completely connected to the hidden layer. From input layer to output layer, a feed-forward propagation algorithm is used. A back-forward propagation algorithm is used in the process of updating weights and parameters.

In the basic back-propagation training algorithm the weights are moved in the direction of the negative gradient, which is the direction of the most rapid performance decrease.

The hidden neurons have wavelet activation functions of different resolutions along with a weight connecting the hidden layer and output layer (ω_i). The weights of synapses of WNN are initialized by

(i) connecting input neurons and wavelet neurons

(ii) connecting wavelet neurons and

(iii) Connecting wavelet neurons and output neurons to small random values (–1 to +1).

The output of input layer is evaluated using wavelet activation function. ωi is the weight connecting the hidden layer and output layer. For an input vector $x = [x_1, x_2, \ldots, x_n]$, the output of the i^{th} wavelet layer neuron is described as follows:

$$\psi_k(x) = \sum_{i=1}^{n} \exp\left(-\left(\frac{x_i - d_k}{t_k}\right)^2\right)\cos\left(5\frac{x_i - d_k}{t_k}\right) \qquad (5.7)$$

where x_i is the i^{th} input vector and k is the number of wavelet nodes, d_k and t_k are translation and dilation parameters respectively. The output of the third layer is the weighted sum of $\psi_k(x)$ (Equation 5.8).

$$y(x) = \sum_{m=1}^{k} \omega_m \psi_m(x) \qquad (5.8)$$

Wavelet network training consists of minimizing the usual least-squares cost function:

$$E = \frac{1}{2}\sum_{j=1}^{s}\left(y_j - o_j\right)^2 \qquad (5.9)$$

where‘s’ is the number of DCT coefficients and o_j is the optimal output of the j^{th} input vector. The learning algorithm of WNN consists of the following steps.

1. Initialize the values for D_i (Scaling vector) and t_i (translation vector)
2. Feed in the input vector X into WNN
3. Calculate the product of hidden layer using Equation (5.10)

$$\psi j(\xi) = \psi(\|Dj(X-tj)\|) \text{ where } j = 1, \ldots, m \qquad (5.10)$$

4. Solve the weight matrix, $W = \psi+\omega$, where $\psi+$ is the pseudo inverse defined as $\psi^{+} = (\psi^{T}\psi)^{-1}\psi^{T}$
5. Obtain the output value of WNN
6. Compare obtained output value with desired output value
7. Calculate cost using Equation (5.33)
8. Repeat Steps (2) to (7) till stopping criterion is met

The learning of the WNN in the above procedure is by the method of solving the pseudo-inverse with fixed parameter initialization. Therefore, only the weight matrix W needs to be adjusted during the training of WNN, in order to map the underlying relationship between the input and output space. The Mean Square error of the difference between the network output and the desired output is calculated. This error is back propagated and the weight synapses of output and input neurons are adjusted. With the updated weights, error is calculated again. Iterations are carried out till the error is less than the tolerance.

- **Support Vector Machine**

The second classifier used is Support Vector Machine, which given a set of input data and predicts, for each given input, which of two possible classes the input is a member. This makes SVM a non-probabilistic binary linear classifier. SVM classifier is well known for its generalization performance and ability to handle high dimensional data. Support Vector Machines were introduced by Vapnik and have proved to be fast effective classifier.

Considering the binary classification case, let $((x_1, y_1) \ldots (x_n, y_n))$ be the training dataset where xi are the feature vectors that represent the observations and $y_i \in (-1, +1)$ be the two labels that each observation can be assigned to. From these observations, SVM builds an optimum hyperplane (a linear discriminant in the kernel transformed higher dimensional feature

space) that maximally separates the two classes by the widest margin by minimizing the following objective function (Figure 5.5).

$$\min_{w,b,\xi_i} \frac{1}{2} w.w^T + C \sum_{i=1}^{N} \xi_i \qquad (5.11)$$

where w is the norm of the hyperplane, b is the offset, $y(x_i)$ are the labels and ξi are the slack variables that permit the non-separable case by allowing misclassification of training instances.

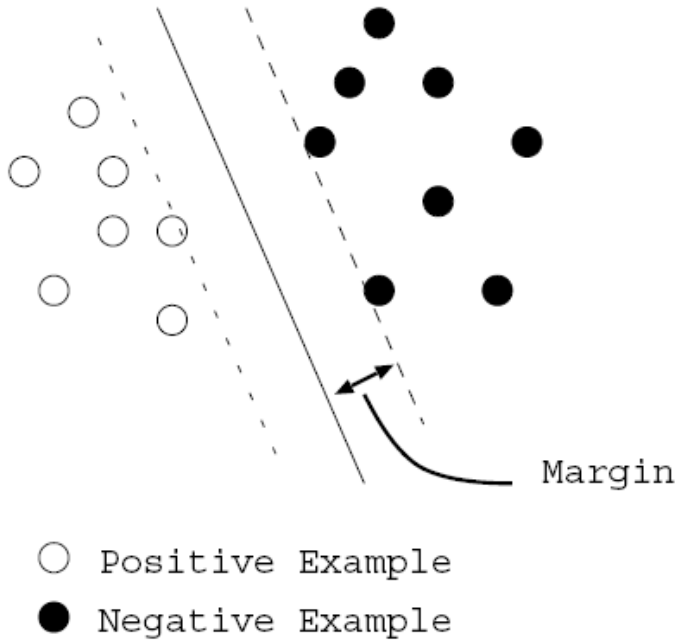

Figure 5.5 : Support Vector Machine Hyperplane

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

Although SVMs were originally designed as binary classifiers, approaches that address a multi-class problem as a single “all-together” optimization problem exist (Weston and Watkins, 1999). A multi-class classification task usually involves separating data into training and testing sets. Each instance in the training set contains one ‘target value" (i.e. class labels) and several

“attributes" (i.e. features). The goal of SVM is to produce a model (based on the training data) which predicts the target values of the test data given only the test data attributes.

SVM Tool is a language independent sequential tagger implemented in Tamil POS tagging. The SVM tool used in this work consists of three components trainer, tagger and evaluator. For SVM learner trains a set of SVM classifiers with different feature combinations using SVM light software in C language. For a two class problem binary SVM is used, if more than two classes binary SVM extended to multiclass SVM (Alexandradis and Zapranis, 2013). The evaluator component evaluates the performance output for the test set. In this paper, pair wise sequential labelling SVM classification method with five degree polynomial kernel function is implemented.

- **Ensemble SVM-WNN Classifier**

The ensemble classifier is created using the hybrid SVM-WNN classifier as base classifier and using Ada boost algorithm to create 20 different base classifiers. Simulated annealing with multiobjective optimizer (Ekbal and Saha, 2013) combined with majority voting aggregation is used during classification. In the hybrid SVM-WNN classifier, the SVM classifier is used as a pre-processing step to reduce the training set to a subset version that contains more critical details in deciding classification boundary. This is accomplished by first extracting support vectors. The actual classification process starts only after this, where the set of support vectors form the training set. WNN is trained using this set, after which the test data is applied to WNN to analyze its efficiency. Thus the hybrid method classification is performed in two steps (Figure 5.6).

- Step 1 : Original feature is used by SVM and support vectors are extracted
- Step 2 : The extracted support vectors and the corresponding actual output values are fed as new training set for the WNN

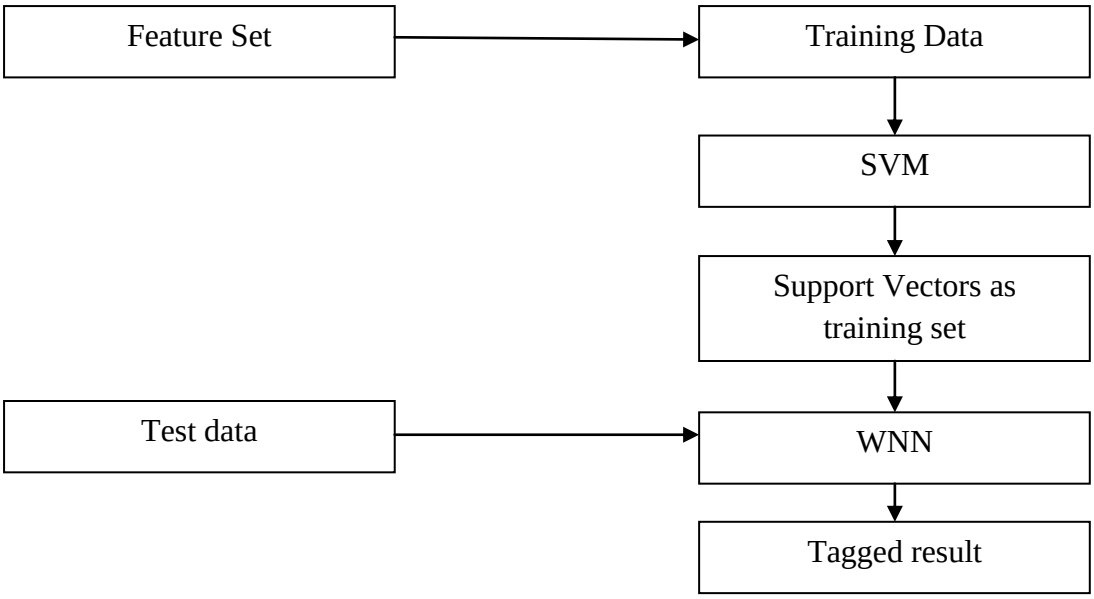

Figure 5.6: Hybrid SVM-WNN Classifier

5.2.2. Chunking

This step organizing information obtained from the previous step into familiar groupings. It is a Natural Language Process that separates and segments sentences into their sub constituents such as noun, verb and prepositional phrases. Examples of chunks include noun phrases, prepositional phrases and verb phrases. Chunking works on POS tagged text, so its accuracy depends upon the accuracy of POS tagger. In this research, as mentioned earlier, a Tool from http://tdil.mit.gov.in (Technology Development for Indian Languages) is used for performing chunking. An example of chunking is shown below.

[அந்த <DET> (B-NP) அழகான <ADJ> (I-NP) பெண் <NN> (I-NP)] NP

5.2.3. Named Entity Recognition (NER)

Named entities include the identification of people names, location and companies / organizations, while digits may include time/date stamp and amount. In a Tamil sentence, the NER identifies words that need to be transliterated, and the remaining words are translated using dictionary. In this research work, this is performed using the tool provided by TDIL (http://tdil.mit.gov.in).

5.2.4. Morphological Analysis

The purpose of a morphological analyzer is to return root word and grammatical information about all the possible word classes for a given word. Morphological analysis phase also includes extraction of the grammatical information including number, gender and tense information for all the tokens. Since Tamil language has a rich inflectional morphology, morphological analyzer is an essential tool during Tamil-English CLTR (Anand kumar *et al*., 2010). For example consider a word "ஓவியங்கள்", which can be meaningfully divided into ஓவியம் (painting) (noun) + கள் (s) (plural), where the first part represents lexical morpheme and second part is a grammatical morpheme. Tamil words have a lexical root followed by one or more affixes. The general framework of morphological analyzer is shown in Figure 5.7.

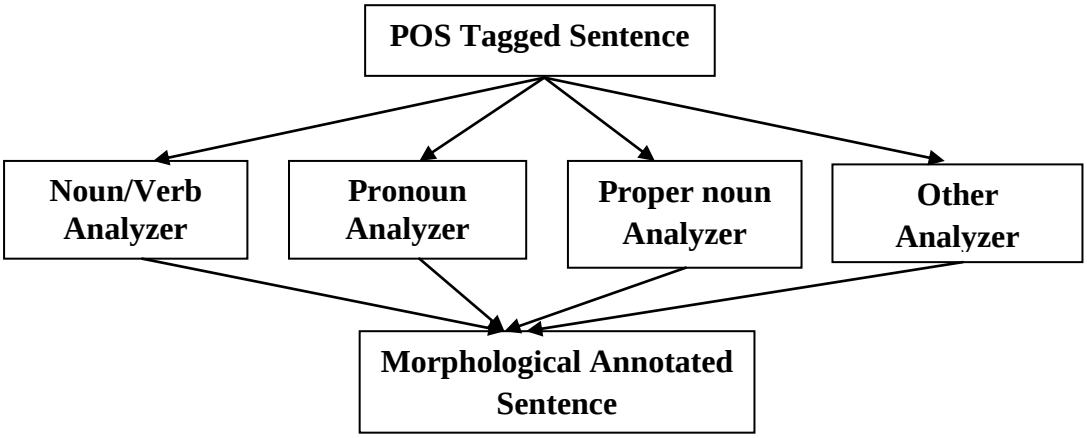

Figure 5.7: General Framework of Morphological Analyzer

Generally rule based approach is used for morphological analyzer and it consists of set of rules for identifying morphemes or root words. If morpheme or words missing in lexicon then the rule based system fails. Therefore a machine learning approach is proposed. In the proposed method, the Tamil noun and verb paradigm classification are done based on the case and tense markers respectively. The possible noun and verb morphemes (Selvam and Natarajan, 2009) are presented in Table 5.3 and Table 5.4 respectively.

TABLE 5.3

CASE SUFFIXES USED WITH NOUN

Cases	Suffixes
Accusative	ஏ, ஐ
Dative	க்கு, ற்கு
Instrumental	ஆல்
Sociative	ஓடு, உடன்
Locative	இல், உள், இடம்
Ablative	இருந்து
Benefactive	க்காக, ற்காக
Genetive	இன், அது, உடைய
Vocative	ஆலே
Clitics	உம், ஓ, தான்
Selective	ஆவது
Interrogative	ஆ

The creation of data is carried out using three steps namely pre-processing which includes Romanization, segmentation, alignment, mapping and bootstrapping. The noun/verb analyzer has two groups of models and it is named as trained model I and trained model II. Model I (segmentation model) trained using sequence of input characters and corresponding output labels. The trained model I or untagged model is used in predicting morpheme boundaries. Model II (morpho-syntactic tagging) is trained using sequence of morphemes and their grammatical categories. The trained Model II or Tagged model is used for predicting morphemes and to assign grammatical class features. The pronoun analyzer works based on pattern matching with the root word. It is a closed set word so it is not tedious to create pronoun root word paradigm. The stemmed pronoun word is directly matched with root word paradigm and it is replaced with corresponding Unicode entry. The remaining part of word is compared with pronoun suffix file list and corresponding entry is replaced. The structure of Pronoun word form is shown in Figure 5.8. The Proper Noun (PN) analyzer compares the word

with the suffix table, if it is present in the table, it strips the suffix. Then the stemmed word is replaced with corresponding root word entry.

TABLE 5.4

CASE SUFFIXES USED WITH VERB

Suffix	Categories	Sub categories	Suffixes
Tense	Present	---	கிறு, கின்று, ஆனின்று
	Past	---	த் , ந் , ற் , இன்
	Future	---	ப் , வ்
Person	First	Singular	ஏன்
		Plural	ஓம்
	Second	Singular	ஆய்
		Plural	ஈர்கள்
		Honorific	ஈர்
	Third	Male Singular	ஆன் ,அன்
		Female Singular	ஆள்
		Common Plural	ஆர்கள்
		Honorific	ஆர் , அர்
		Neutral Singular	அது
		Neutral Plural	அன
Others	Causative	---	இ
	Verbal Noun Untensed	---	அல்
	Infinitive	---	உ
	Imperative	Plural	உங்கள்
		Negative	ஆதே, ஆது
	Passive	---	படு
	Future	Negative	மாட், இல்லை
	Optative	---	முடியும், வேணடும், கூடும், ஆம
		negative	முடியாது, கூடாது, வேண்டாம்
	Sandhi	---	ந, க, ம, ச, த
	Plural	---	கள்

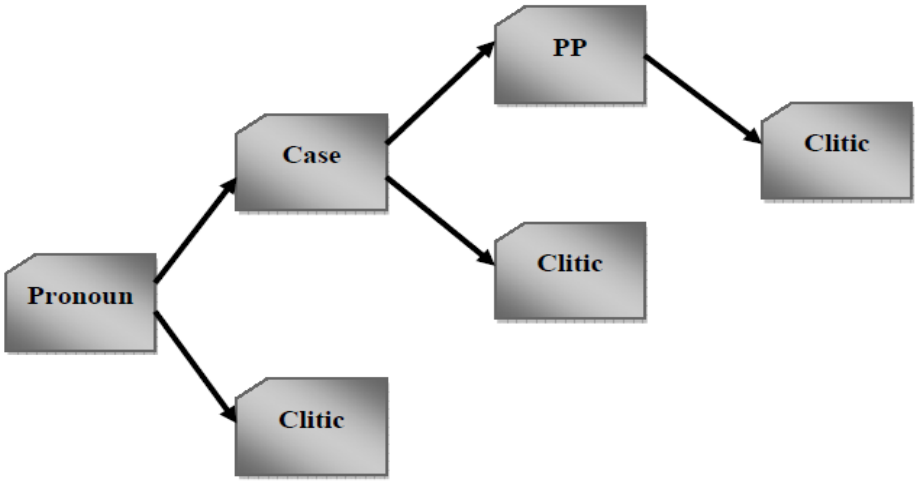

Figure 5.8: Structure of Pronoun word form

The morphological classification problem is solved using sequence labelling approach and training is based on non-linear manner of ensemble SVM classifier and the classifier is called as Ensemble SVM-based Morphological Analyzer model. The homogeneous Ensemble SVM based Morphological Analyzer (ESMA) has single classification methodology, but it takes different form of feature vectors. The ESMA uses Adaboost subspace selection algorithm to train the SVM component classifier and for combining the results. The feature vectors used for Model-I and Model-II are sequence of input characters and sequence of morphemes. The partitioning method, aggregation method and training types are used, similar to the methods explained in Phase I of the research work.

For example, POS tagged Tamil sentence given as input to morphological Analyzer and their results are shown below.

Tamil Sentence: அவன் அலுவலகத்தை நோக்கி நடந்தான்

POS Tagger : <NNP> <CL> <IN> <VM>
<Proper noun> <Common Noun><postposition><Verb finite>

Morphological Analyzer: <Proper Noun (PN)> <Noun> <Other> <Verb>

MA Results :அவன்<root_word>(NNP)

அலுவலக<root_word><CL>+த்(sandhi)+ஐ (Accusative case)

நோக்கி <IN>

நட<root_word><VM>+ந்(த்)(sandhi)+ ஆன்(3Male-Singular)

5.2.5. Word Sense Disambiguation

Word sense Disambiguation is a process of assigning the most appropriate sense for each occurrence of an ambiguous word in a sentence (Sharma and Niranjan, 2012). WSD is an important for applications such as Machine translation and Information retrieval. For example consider a Tamil sentence

Sentence1: என் கேள்விக்கு **விடை** சொல்

Sentence 2: ராமு வீட்டில் இருந்து **விடை**ப் பெற்றான்.

In this example, an ambiguous word is "**விடை"** which has atleast two possible senses, i.e., answer and relieved. The good quality of translation can only be achieved by choosing a right sense of an ambiguous word, and this process of identifying a correct sense for a word is done using WSD procedure. The WSD algorithm disambiguates the ambiguous word based on their context or collocations. Collocations are the words that are adjacent to a target word, strongly indicating the sense of an ambiguous word.

Basically there are three types of lexical ambiguities they are (i)Polysemy (ii)Homonymy and (iii)categorical ambiguity (Hrist, 1987).

- Polysemy ambiguity

It is a word or phrase with different, but with related senses and it occur in the form of both noun and verb POS categories. The word " **பிடி**" is one of the most ambiguous polysemous word, and it has several senses such as capture, shape, catch and massage.

- Homonymy ambiguity

The word or phrase having multiple meaning, but they are of unrelated senses. Homonymous words have senses that are clearly distinct unlike the case of polysemous words. The Tamil word "**மாலை**" is homonymous word with two different senses such as evening and garland.

- Categorical ambiguity

The word or phrase having multiple meaning, but they are of different grammatical categories called categorical ambiguity. For example, the Tamil word "ஓடு" has distinct senses with different POS categories such as Tile (Noun) and run (verb).

The related works of WSD uses dictionary (Lesk, 1986; Dagan and Itai, 1994) or thesaurus (Yarowsky, 1992; Yaroswsky, 1995) or sense tagged corpus (Ng and Lee, 1996) for disambiguation. In earlier works, collocations are extracted from dictionary or thesaurus or sometimes given manually to bootstrap algorithm is a time consuming process and painstaking effort (Baskaran and Vaidehi, 2004). The proposed method called enhanced WSD with Part-of-speech and Clustering based Sense-collocation (WSDPCS) procedure solve the earlier issue, by automating the process of collocation extraction using clustering technique, and disambiguation of the word sense is carried out in a two step manner that are pointed below. The focus of this WSDPCS approach is to disambiguate homonymous and categorical ambiguity Tamil words.

(i) POS tagging in disambiguating Word Senses.

(ii) Enhanced with Clustering and Sense-collocation dictionary based disambiguation.

- **POS tagging in disambiguating Word Senses**

The first step is to disambiguate an ambiguous Tamil word directly by applying the POS tagger. The senses of ambiguous word having same POS grammatical category labels are handled using next step. The categorical ambiguity words are disambiguating using Tamil and English POS tagger. The process of disambiguating word senses using POS tagging is shown in Figure 5.9. The input Tamil word is tagged using Tamil POS tagger and it is given for translation to identify possible English word senses. Then the word senses are tagged using English POS tagger, and tags of both ambiguous word and corresponding sense words are compared. Finally which sense tag matches with an ambiguous tag that type of sense is assigned to an ambiguous word.

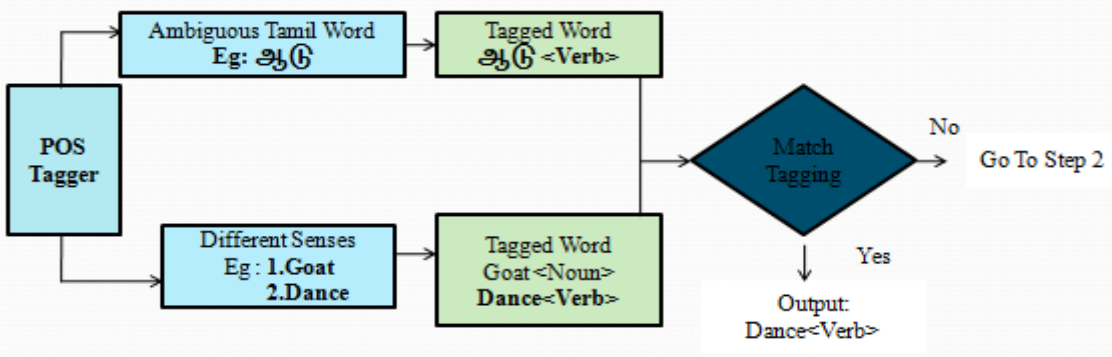

Figure 5.9: Process of disambiguating word senses using POS tagging

- **Enhanced with Clustering and Sense-collocation dictionary based disambiguation**

The section consists of two phases that are training phase and testing phase. The Architecture of WSD uses clustering and sense-collocation dictionary is presented in Figure 5.10. Initially in Tamil document corpus, stop words are removed and ambiguous word list is created. Collocations are context words that appear on either side of an ambiguous word till specified window size, and different window sizes are used by different researches. In this chapter, size of sliding window is 25, i.e. context words of 25 are extracted before and after of an ambiguous word. All the context words of an ambiguous word are collected from the document corpus to create a Context space which is denoted as S and it is shown in Fig 5.11. Context Space S consists of context vectors, and these context words are morphological analyzed to return root words and these are included in context word list.

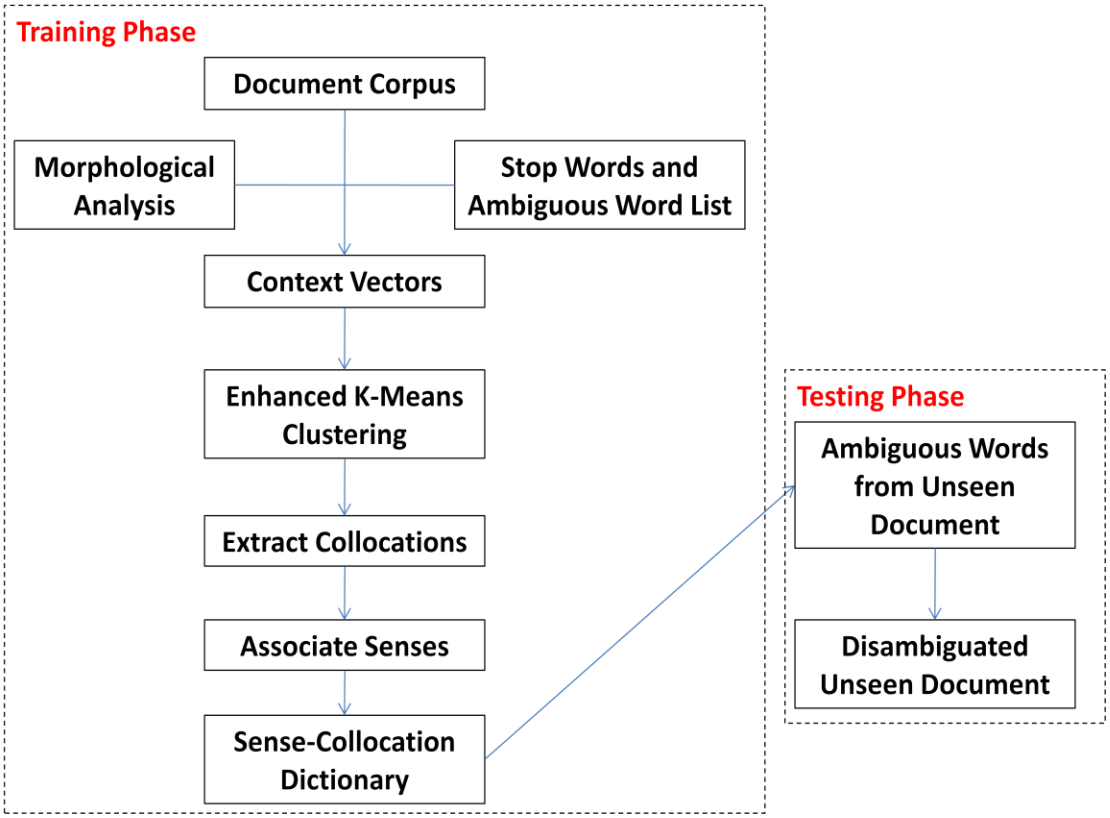

Figure 5.10: Architecture of WSD using clustering and sense-collocation dictionary

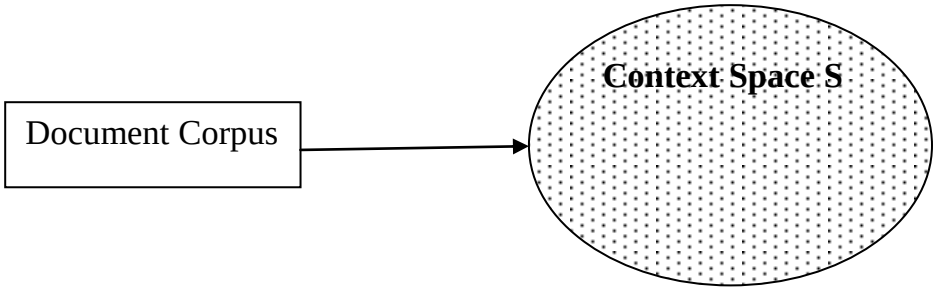

Figure 5.11: Representation of Context Space

The next step in training phase is Enhanced K-means clustering that produces k-final clusters. The performance of conventional K-Means algorithm depends on the selection of K value. In conventional algorithm, this value is selected by user and when inappropriately selected, degrades clustering performance. To solve the above issue of selection of optimal K value, an ensemble technique is used. Enhanced K –means algorithm generates different k cluster centers ranging between 2 to 30. To find an optimal K value and optimal clustering set,

majority voting algorithm is implemented. The selection of optimal K-value is embedded during the clustering process, and this automated process saves search time while considering large number of ambiguous words.

At the end of clustering, k-final clusters are created and the collocations present in these clusters are contextually similar. Before extracting the collocations from clusters, identification of potential seed words in cluster is an important task. In each cluster, collocations are ranked according to the high priority based on the log-likelihood ratio (yarowsky, 1995). The collocations are extracted from the clusters are assigned with right senses are done using human-annotators. Finally sense-collocation dictionary is constructed having attributes such as ambiguous words, major sense and top collocations. The words that cannot be disambiguated using step1, is handled using sense-collocation dictionary. The sample entries in sense-collocation dictionary are shown in Table 5.5.

TABLE 5.5

SAMPLE ENTRIES IN SENSE-COLLOCATION DICTIONARY

Ambiguous Words	Sense	Top collocations
மாலை	Evening	**காலை**, **பகல்**, **சூரியன்**, **நாள்**,**மாதம்**, **இரவு** **மதியம்**, **நேரம்**, **விடியல்**.
மாலை	Garland	**பூ**, **மணம்**, **ரோஜா**, **மல்லிகை**, **பூஜை**, **நறுமணம்** **மகளிர்**, **பூங்கொத்து**
நூல்	Book	**படி**,**ஆசிரியர்**, **படிபகம்**, **அச்சு**, **பக்கம்** **தாள்**, **பரீட்சை**
நூல்	thread	**ஊசி**,**தறி**, **சாயம்**, **இலை**, **கதர்**, **நெசவு**, **உடுத்து**, **வண்ணம்**, **துணி**

5.3. TRANSLATION

The next step after pre-processing is translation and it is carried out using knowledge sources and rules set. The output of morphological analyzer is root words and its grammatical morpheme. The root words are directly translated using bi-lingual dictionary. Sometimes several words may not found in the bi-lingual dictionary called Out-of-Vocabulary words. The problem of OOV is solved in proposed HMT based system with the help of semantic dictionary. If a word

is not found in the bi-lingual dictionary, then it is searched in semantic dictionary to obtain an equivalent Tamil word. The semantic equivalent word of the OOV word is translated using root word dictionary. A single word may have several translations, and this ambiguity problem is handled using word sense collocation dictionary. WSD procedure helps in choosing the best hypothesis translation from all possible translations. The remaining part of word is grammatical categories which are translated by applying tense marker and gerund ending rules and some of rules are presented in Figure 5.12.

- If Pronoun (PRP) =" அவன் /அவள்" and VBP="கிறான்/கிறாள்", then add **"is"** after it.
- If Pronoun (PRP) =" அவர்" and VBP="கிறார்கள்", then add **"are"** after it.
- If Pronoun (PRP) =" அவன்" and VBP=" ந்தான்", then add **"was"** after it.
- If Pronoun (PRP) =" அவள்" and VBP=" ந்தாள்", then add **"was"** after it.
- If (PRP) = "அது " and VBD=" கிறது " , then add **"is"** after it.
- If (PRP) = "அது " and VBD=" ந்தது ", then add **"was"** after it.
- If (PRP) = "அவர் " and VBD=" ந்தார்கள் " , then add **"were"** after it.
- If Personal Pronoun (PP) = "நான் " and VBP=" கிறேன்" then add "**am**" after it.
- If (PP) = "நாம் "and VBP= "கிறோம்" then add "**are**" after it.
- If (PP) = "நான் " and VBP= "ந்தேன் " then add **"was"** after it
- If (PP) =" நாம்" and VBP= "ந்தோம்" then add "**were**" after it
- If TO=" க்கு" then add **"to"** before the noun.
- If (PRP) = "அவன்/ அவள் / அவர்/ நான் / நாம்" and MD= "வான்/ வாள்/ வார்கள்/ வேன் / வோம் " then add "will / shall / Could/ Should "respectively.
- If Noun/PRP + ஆல் then add **"with"** before it.
- If Noun/PRP + ஒடு then add **"with"** before it.
- If Noun/PRP + இன் then add **"of"** before it.

Figure 5.12: Tense marker and gerund ending rules

The knowledge based resources such as bilingual dictionary and semantic dictionary are obtained from TDIL (http://tdil.mit.gov.in), and also it collected from various sources of Internet. The transliteration is carried out in next section; as a result translated English sentence is obtained. Tamil language mostly follows Subject-Object-Verb (SOV) pattern, where as English language is a Subject-Verb-Object (SVO) pattern. The re-arrangement of Tamil words into the correct structure of English language is done using Rule-based reordering. To rearrange simple sentences from Tamil to English language common reordering rule is applied that is presented below. Finally tagged words are rearranged according to the correct structure of English language.

Re-ordering rule from Tamil to English language: INJ(Interjection) /PP(Personal pronoun) /WP(Wh-pronoun) / WRB(Wh-adverb) / WDT(Wh-determiner) / DT(determiner) / NNP (Proper noun) / PRP (pronoun) / MD(modal) /VBZ (verb present part) /VBP (verb present) /VBN (Verb past part) /VBZ (verb present) /VBD(Verb past) /VB(verb) /CC (conjunction) /RB (Adverb) /JJ (Adjective) /JJR (Adj-Comparative) /JJS (Adjective-Superlative) /IN (preposition)/TO (to)/NN (Noun)/NNS (Noun plural).

5.4. TRANSLITERATION WITH ERROR CORRECTION

Transliteration is task of converting one form of script to another form of script. The words that cannot be translated using dictionary are named entities. The named entity recognizers identify named entities, and give the entities as input to the transliteration engine. However proper nouns and common nouns are very often appear in transliterated forms which play an important role in retrieval of documents. Transliteration is first performed to convert named entities and numbers. The first pass retrieval carried out using a character transformation procedure which converts each Tamil character to its English equivalent. For this purpose, a Tamil-English Character Mapping Table (http://www.azhagi.com/az-tamil-modern.html) is used.

During second pass retrieval, statistical transliteration model (Saravanan *et al*, 2011) is implemented that hypothesizes a match between named entity term and a document term in the "comparable" document pair of top 30 retrieval documents. It is an extension of W-HMM word alignment model that makes use of a both the transition and emission models in richer context compared to the classic HMM model.

5.5. CHAPTER SUMMARY

This chapter presented the steps involved during query translation of Tamil query into English query. To achieve good quality translation of translation, hybrid MT based architecture is proposed. Hybrid MT combines rule based approach and statistical knowledge based approach. The rule based approach was used during Translation and the statistical method was used for Transliteration. After translating the query, the next step to be performed is the actual retrieval of relevant documents. This is the primary focus of Phase III of the research work

focuses and the algorithm description of the proposed retrieval algorithm is presented in the next chapter, Chapter 6, **Design of Text Retrieval System.**