# Word Sense Disambiguation using case based Approach with Minimal Features Set

Tamilselvi P[*]
Research Scholar, Sathyabama Universtiy,
Chennai, TN, India
Tamil_n_selvi@yahoo.co.in

S.K.Srivatsa
St.Joseph College of Engineering
Chennai, TN, India

**Abstract:**

In this paper we presented a case based approach for word sense disambiguation using minimal features set. To make the disambiguation, we took only two features for two different methods, post-bigram (immediate left word with ambiguous word – $l_1w$) and pre-bigram (ambiguous word with immediate right word of it – $wr_1$). To classify the cases for disambiguation, we followed three steps: instance or case identification from the distributed dictionaries, instance filtering based on the PoS and finally, case selection based on similarity measuring methods. Here, we used three different distant measuring functions, Euclidean, city-block and cosine methods. Set of reduced form of cases are treated as input for disambiguation. For making disambiguation, we applied K-nearest neighboring algorithm and artificial Neural Network. Among these two, KNN produced better disambiguation accuracy of 81.75% from cosine cases using pre-bigram features.

**Keywords:** Word Sense Disambiguation, Pre-bigram, Post-bigram, similarity function, case based reasoning (CBR), Part-of-Speech (PoS), K-Nearest Neighboring method (KNN), Artificial Neural Network (ANN) .

## 1. Introduction

One important problem of Natural Language Processing is figuring out what a word means when it is used in a particular context. The different meaning of a word is listed as its various senses in a dictionary. The task of word sense disambiguation is to identify the correct sense of a word in context. Improvement in the accuracy of identifying the correct word sense will result in better machine translation systems, information retrieval systems, etc. We adopted case based reasoning (CBR) for sense disambiguation and here, text comparison is required to disambiguate the words in the input.

The aim of CBR is to reuse solutions of similar cases to solve the problem at hand [Weber, R.O., Ashley, K.D, 2006]. Clearly, the ability to compare text content is vital in order to identify the set of relevant cases for solution reuse. However a key challenge with text is variability in vocabulary which manifests as lexical ambiguities such as the polysemy and synonymy problems [Simpson, G.B, 1984].

## 2. Related Work:

Case-based Reasoning is an approach in artificial intelligence that differs from other artificial intelligence approach. [Krisda Khankasikam, 2011] used case based reasoning to get solution (meta data) for the current situation. If the existing cases failed to produce the solution, case adaption was done to make changes in existing cases, in such a way to produce solution for the situation. To identify the similarity between the case and the input data, Euclidean distance was used.

Common problems faced in natural language processing are data sparseness and inconsistency in vocabulary [Juan A. Recio-Garcia and Nirmalie Wiratunga, 2010]. They used case based reasoning to infer knowledge from web pages. Cases were defined as a pair of problem-solution along with the vocabularies. Frequency of a term in a case was also calculated to group the content based on the context. It is stated that to measure the similarity between the cases and input, any one of the similarity function, namely, Euclidean, Cosine or KL-Divergence was used.

To avoid data sparseness, we tried to make disambiguation with minimum features, that is, PoS of two consequent morphologically individual (non-compound) words. Inconsistency is a common problem of any language. To uncover the inconsistency, word sense disambiguation algorithms are applied by considering the neighboring words in the sentences or frequent words in the document. To solve ambiguity of a word, supervised and unsupervised approaches can be applied [Yarowsky, 1992]. Former approach uses large annotated datasets mainly for training purpose and the later does not require any training to disambiguate, so it is defined without any annotated dataset.

[Pedersen, 2001] experimented the use of bigrams for WSD with a decision tree and naive Bayes classifier. He tested different bigrams that occur close to the ambiguous words (within approximately 50 words to the left or right of the ambiguous word) as possible disambiguation features. He applied statistical method to disambiguate texts using decision tree with bigram concept.

[Zhimao Lu, Ting Liu and Sheng Li, 2004] extracted mutual Information (MI) of the words as input vectors for back-propagation neural network. The network is tested with maximum feature sets varying from ten words from left and ten from right with respect to ambiguous word. When the number of features increases, the sparseness is unavoidable. Smoothing is really required to overcome the above problem for improving the performance.

We proposed a system to disambiguate words using case based reasoning with minimal features. With CBR, our aim is to get the solution by comparing the current problem description with a set of past cases maintained in a casebases, referred as distributed E-dictionaries [P.Tamilselvi, S.K Srivatsa, 2009]. Rest of the paper is organized as follows: section 3 provides about the system architecture, section 4 about the experimental results and in section 5 the conclusion.

## 3. System Architecture

To find the sense of a word by removing its ambiguity, we followed three major steps: pre-disambiguation process, case extraction and disambiguation process

### *3.1. Pre-Disambiguation Process:*

Consider, the input sentence S with $W_i$ (i=1…n) words, represented as in equation 1.

$$S = \sum_{i=1}^{n} W_i, i = 1 \dots n \qquad (1)$$

In pre-disambiguation process, the given input S is tokenized and the compound word like '*took_place*' are separated as '*took*' and '*place*'. Now, the decomposed sentence DS will have $w_j$, m>=n, j=1…m, represented in equation 2.

$$DS = \sum_{j=1}^{m} w_j, m \geq n, j = 1 \dots m \qquad (2)$$

Next, the morphological form of the word is extracted, i.e morphological form of the individual word 'took' extracted as 'take' and its PoS tag 'VB' (verb) is also attached by Hidden Markov Model parsing technique [P.Tamilselvi, S.K.Srivatsa, 2010 (J)]. Ambiguous words in the input sentence are isolated using WordNet [P.Tamilselvi, S.K.Srivatsa, 2010 (C)] and reformed into two different vector forms: pre-bigram ($T_1$) and post-bigram ($T_2$).

### *3.2. Case Extraction:*

#### *3.2.1. Case Representation:*

Structure of cases in the distributed E-dictionaries is as in table-1. Here, $lw_3$, $lw_2$, and $lw_1$ are left most words positioned from ambiguous word and similarly, $rw_1$, $rw_2$ and $rw_3$ are right most words positioned from ambiguous word, totally, seven word information are in the dictionaries (3L (left words) + W + 3R (right words)). 3L and 3R are referred as semantic marks of the ambiguous word. To frame a vector, semantic markers along with ambiguous words are needed to transfer into numeric form. These words are morphologically individual words having their individual numerical values (varying from .1 to .17) based on their PoS. For bigram, only two elements are taken out of these seven elements, $Lw_1$ + W (post-bigram) and W + $rw_1$ (Pre-bigram). In the same way, ambiguous words in the input sentence should also be refined with post-bigram and pre-bigram vector values. Since it is a bigram, each row vector is defined with only two features (2 columns).

Table-1: Structure of Cases in distributed E-Dictionaries

| word | sense | Sense-tag | $lw_3$ | $lw_2$ | $Lw_1$ | ambiguous word (w) | $rw_1$ | $rw_2$ | $rw_3$ |
|---|---|---|---|---|---|---|---|---|---|

*3.2.2. Case Selection:*

Case extraction is done in three steps: instance or case identification, instance filtering and finally, instance selection. In case identification, the specific distributed E-dictionary having ambiguous word is chosen and cases with the ambiguous words are selected, referred as case identification. From the identified cases, cases with other PoS are discarded, i.e cases with ambiguous word's Pos are retained for disambiguation process, and it is referred as case filtration. Finally, similar cases are selected using similarity measuring functions such as Euclidean, cityblock and cosine functions. Outcome of these are passed as input for disambiguation process.

*3.2.3 Similarity Measuring Functions:*

- Euclidean Function [website1]:

The Euclidean distance between points p and q is the length of the line segment connecting them. In Cartesian coordinates, if p = ($p_1$, $p_2$,..., $p_n$) and q = ($q_1$, $q_2$,..., $q_n$) are two points in Euclidean *n*-space, then the distance from p to q, or from q to p is given by:

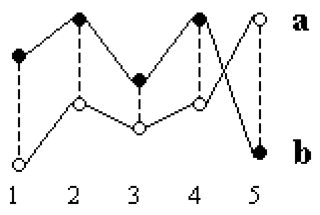
$$d(p,q) = d(q,p) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \qquad \textbf{(3)}$$

- *Cityblock Function [website2]:*

The *City block distance* (also referred to as *Manhattan distance* ) between two points, *a* and *b*, with *k* dimensions is calculated as:

$$d(a,b) = \sum_{j=1}^{k} |a_j - b_j| \qquad (4)$$

The *City block distance* is always greater than or equal to zero. The measurement would be zero for identical points and high for points that show little similarity. The figure below shows an example of two points called *a* and *b*. Each point is described by five values. The dotted lines in the figure are the distances ($a_1$-$b_1$), ($a_2$-$b_2$), ($a_3$-$b_3$), ($a_4$-$b_4$) and ($a_5$-$b_5$) which are entered in the equation above.



In most cases, this distance measure yields results similar to the *Euclidean distance*. Note, however, that with *City block distance*, the effect of a large difference in a single dimension is dampened (since the distances are not squared).

- Cosine Similarity Function [website3]:

This cosine similarity is used as a similarity measure between any two vectors representing documents, queries, snippets or combination of these. Cosine similarity is calculated from the formula:

$$Sim(A,B) = cosine\,\theta = \frac{A.B}{|A||B|} = \frac{x_1 * x_2 + y_1 * y_2}{(x_1^2 + y_1^2)^{1/2}(x_2^2 + y_2^2)^{1/2}} \qquad (5)$$

**3.3. Disambiguation Process**

Disambiguation is one of the main tasks for Natural Language Processing. Different ways of techniques such as Statistical approach [Marine Carpuat, Dekai Wu, 2008], decision tree [Pedersen, 2001], and artificial neural network [Zhimao Lu, Ting Liu and Sheng Li, 2004] etc are used for disambiguation. We took up case based disambiguation. From the cases, to select the correct situation or case (solution) for the current situation or for the ambiguous word in the input (problem) we used two methods, k-nearest neighboring (KNN) and Artificial Neural Network (ANN). Both are used with two different feature sets, T1 (Pre-bigram) and $T_2$ (Post-bigram). Disambiguation solution is derived from K-nearest neighbor (KNN) with k=1.

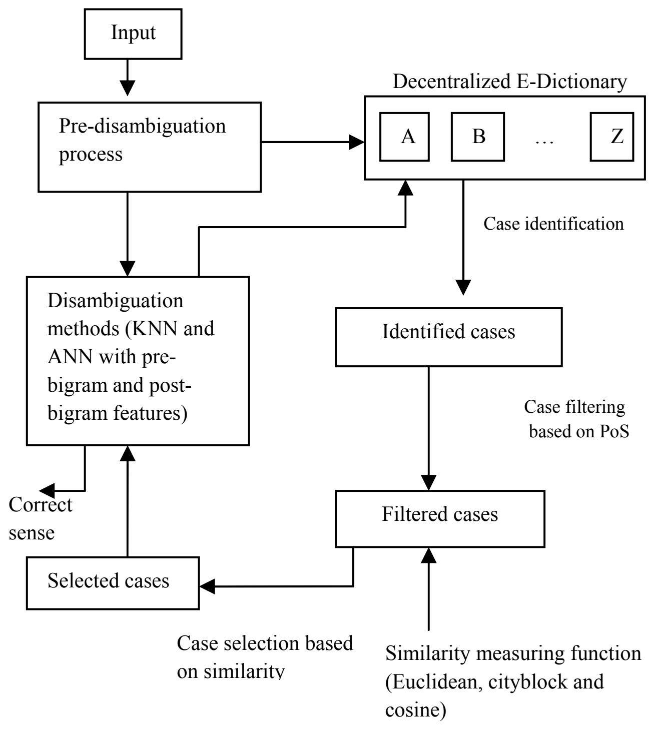

Input
Pre-disambiguation process
Decentralized E-Dictionary
A
B
…
Z
Case identification
Disambiguation methods (KNN and ANN with pre-bigram and post-bigram features)
Identified cases
Case filtering based on PoS
Correct sense
Filtered cases
Selected cases
Case selection based on similarity
Similarity measuring function (Euclidean, cityblock and cosine)

Fig-1: System Architecture

With ANN, cascade feed forward back propagation network with one hidden layer (ANN) (network architecture given in Fig-2) is used for disambiguation. Tangent Sigmoid transfer function is applied in hidden layer and liner transfer function is used in output layer. Levenberg-Marquardt back propagation function is used for training. Gradient decent with momentum weight and bias learning function is used for learning. To measure the performance, mean squared error function (***mse)*** is used. The network is adopted and trained by changing the weights repeatedly for producing better result.

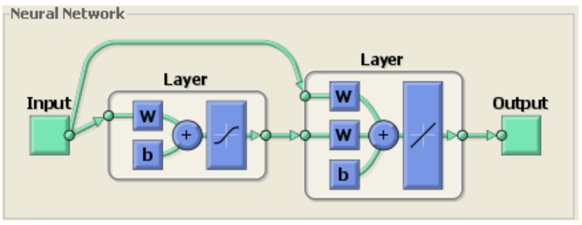


Fig-2: Neural Network Architecture

## 4. Experimental Results:

Interpreted sentences from Brown Corpus [P.Tamilselvi, S.K Srivatsa, 2010 (C)] are considered for our work. Totally, 1500 sentences are taken and 80% (1200) of sentences are treated as cases and remaining 20% (300) are taken for testing. We used three different similarity-finding functions for case selection. After collecting the similar cases by the functions, $C_i$ (i=1,2,3), KNN is applied with k=1, that is, the first minimal distant case is treated as expected output for the current situation. If tie exists between cases, best case will be selected on random basis. Same set of cases, $C_i$ is taken as training data for the neural network. After completion of training, the feature vector of the ambiguous word in the input sentence is simulated with the network to get the relevant output. Disambiguation accuracy of two different methods (KNN & ANN) on three different similarity cases $C_i$, (i=1,2,3) with two different feature vectors ($T_1$ & $T_2$) is given in chart-1.

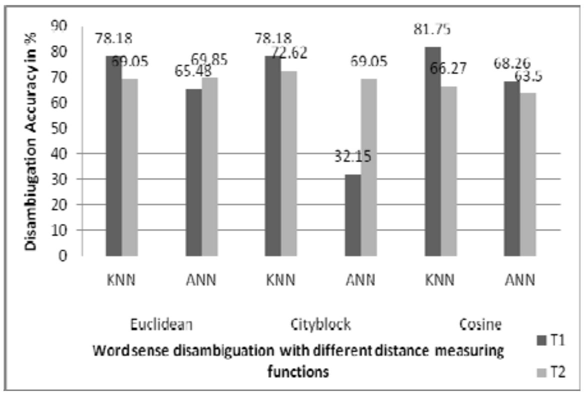


Chart-1: Disambiguation Accuracy

In the chart, T1 and T2 represent pre-bigram and post-bigram feature vectors. The chart explicitly shows that overall performance of KNN is looking good when compared to artificial neural network. Among all the similarity functions, cosine cases with KNN with pre-bigram produced better accuracy having 81.75%, nearly, 82%.

Input sentences are considered into two segments, sentences having less than or equal to 10 words (seg1) and having more than 10 words (seg2). From the test, we observed that, the disambiguation accuracy produced by KNN with cosine cases with T1 and T2 on both segments seg1 and seg2 are not having much difference. Both are producing almost equal to 82% - 83% accuracy. But KNN with T1 and T2 on Euclidean and cityblock cases of seg1 produced accuracy as 83% as like in cosine cases, but with seg2, accuracy level of T1 is 78% and T2 is lesser than T1. But, when seeing the outcome of ANN, seg2 with T2 produced better result, still KNN gives more satisfactory outcome on accuracy than ANN, shown in table-2.

Table-2: Performance comparison based on size of the sentences

| Similarity Function | Disambiguation Methods | Seg1(≤10) | | Seg2 (>10) | |
|---|---|---|---|---|---|
| | | T1 | T2 | T1 | T2 |
| Euclidean $C_1$ | KNN | 83 | 83 | 78 | 69 |
| | ANN | 58 | 67 | 65 | 70 |
| Cityblock $C_2$ | KNN | 83 | 83 | 78 | 73 |
| | ANN | 58 | 67 | 62 | 69 |
| Cosine $C_3$ | KNN | 83 | 83 | 82 | 66 |
| | ANN | 58 | 67 | 68 | 69 |

We also tried to measure the performance of disambiguation based on disambiguation time. For this, we used the basic command tic and toc (stop watch counter) to measure the processing time of KNN and ANN with T1 and T2 on $C_i$(i=1,2,3), with an assumption that no other task should be assigned to CPU. Since the time given by tic-toc is not constant always, we did each disambiguation process for five times (no changes in output, only the time values got changed) and average of that is treated as processing time, given in table-3.

Eculidean($C_1$)-KNN-T1 took least processing time (0.01046 seconds), but, the overall accuracy of it, is only 78.18%, (Chart-1). For cosine($C_3$)-KNN-T1, it took some more seconds (0.005789 seconds) to complete the process, but, its accuracy level is 81.75% (chart-1). Even thought, it took more processing time than Eculidean($C_1$)-KNN-T1, because of its accuracy level with minimal features, we recommended this for disambiguation process, among all the methods we tested.

Table-3: Performance based on processing time

| Similarity Function | Disambiguation method | Processing Time in Seconds | |
|---|---|---|---|
| | | T1 | T2 |
| Euclidean $C_1$ | KNN | 0.01046 | 0.006246 |
| | ANN | 0.744695 | 0.499625 |
| Cityblock $C_2$ | KNN | 0.012162 | 0.005652 |
| | ANN | 0.779691 | 0.589325 |
| Cosine $C_3$ | KNN | 0.005789 | 0.005381 |
| | ANN | 0.638289 | 0.610282 |

## 5. Conclusion

We used three different similarity functions Euclidean, Cityblock and Cosine for case selection from distributed E-dictionaries. Cases are extracted based on Pre-bigram (ambiguous word + immediate next word) and Post-bigram (preceding word + ambiguous word), with only two feature elements in each row vector. Cases are processed with KNN and ANN for the ambiguous words in the list (prepared from input sentence). Among these, cases selected by Cosine angle function with Pre-bigram vectored KNN produced 81.75% disambiguation accuracy with all types of segments (seg1 and seg2) of sentences. Level of accuracy performance can be increased by raising the feature elements size as three or four in the row vector.

## References:

[1] Juan A. Recio-Garcia 1 and Nirmalie Wiratunga, (2010), Taxonomic Semantic Indexing for Textual Case-Based Reasoning, ICCBR'2010. pp.302-316

[2] Krisda Khankasikam, (2011), Metadata Extraction Using Case-based Reasoning for Heterogeneous Thai Documents, International Journal of Computer and Electrical Engineering, Vol.3, No.1, February, 2011

[3] Marine CARPUAT Dekai WU, (2008), Evaluating the Word Sense Disambiguation Performance of Statistical Machine Translation, LREC 2008

[4] P.Tamilselvi, S.K.Srivatsa, (2009) Decentralized E-Dictionary (DED) for NLP task, Proceedings of ICMCS International conference on Mathematics and computer Science, India, 2009

[5] P.Tamilselvi, S.K.Srivatsa, (2010) (J), Part-Of-Speech Tag Assignment Using Hidden Markov Model, International Journal of Highly reliable Electronic System, Vol-3, No-2, 2010.

[6] P.Tamilselvi, S.K.Srivatsa, (2010) (C), A Study on Lexicographical Information using open source lexical databases, Proceeding of NCRTCSE National conference on Recent Trends in Computer Science and Engineering, 2010

[7] Simpson, G.B, (1984), Lexical ambiguity and its role in models of word recognition. Psychological Bulletin 92(2), 316–340

[8] T. Pedersen, (2001), A decision tree of bigrams is an accurate predictor of word senses, in: Presented at Second Annual Meeting of the North American Chapter of the Association for Computational Linguistics, 2001.

[9] Weber, R.O., Ashley, K.D., Br¨uninghaus, S, (2006), Textual case-based reasoning. The Knowledge Engineering Review 20(03), 255–260.

[10] Yarowsky. D,, (1992), Word sense disambiguation using statistical models of Roget's categories trained on large corpora, In: Zampolli, A., ed. Computatuion Linguistic'92. Nantas: Association for computational Linguistis, 1992, 454-460.

[11] Zhimao Lu, Ting Liu, and Sheng Li. (2004), Combining neural networks and statistics for chinese word sense disambiguation. In Oliver Streiter and Qin Lu, editors, *ACL SIGHAN Workshop 2004*, pages 49-56.

[12] (website1) http://en.wikipedia.org/wiki/Euclidean_distance

[13] (website2) http://stn.spotfire.com/spotfire_client_help/hc/hc_city_block_distance.htm

[14] (website3) http://www.miislita.com/information-retrieval-tutorial/cosine-similarity-tutorial.html