Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 30 (2012) 546 – 553

Procedia Engineering

www.elsevier.com/locate/procedia

International Conference on Communication Technology and System Design 2011

Optimal Word Sense Disambiguation with Minimal Feature Set using Neural Network

Tamilselvi P[a], S.K.Srivatsa[b],a*

[a]*Sathyabama Universtiy, Chennai, TN, India*
[b]*St.Joseph College of Engineering, Chennai, TN, India*

Abstract

In this paper, a supervised approach for word sense disambiguation using neural network with minimal feature sets was implemented. Three different networks represented with one hidden layer in which the hidden neurons ranging from 5 to 20 with the increase of 5 neurons at a time are constructed for disambiguation. Disambiguation is tried with minimum two features, bigram and maximum three features, trigram. Number of input for the network is based on the number of features taken for disambiguation process. Bigram takes only two features including ambiguous word and trigram takes only three features (including ambiguous word). Performance is measured using four different error functions. Out of 60 different network architecture, In-trigram based pattern recognition network with 20 neurons produced outstanding performance with 85.72% accuracy.

© 2011 Published by Elsevier Ltd. Selection and/or peer-review under responsibility of ICCTSD 2011
Open access under CC BY-NC-ND license.

Keywords: Word Sense Disambiguation; Bigram; Trigram;Neural Network; Part of Speech (PoS); Performance measuring functions

1. Introduction:

One major problem of Natural language processing (NLP) is figuring out what a word means when it is used in a particular context. The different meanings of a word are listed with 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. Many research on word sense disambiguation has

* Tamilselvi.P : Mob +91 9840494754
E-Mail Address: nagas_67@hotmail.com

1877-7058 © 2011 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
doi:10.1016/j.proeng.2012.01.897

made it known that several text features such as surrounding words, part-of-speech, collocations, verb-object relations, text position etc. can contribute to solve the lexical ambiguity, Michael [5], A. Azzini et al [2], Massimiliano et al [6], Ng and Zelle.[4]. In this paper, disambiguation is tried with only part-of-speech of either two words or three words (including ambiguous word).

Disambiguation approach can be classified into supervised and unsupervised, based on the sentences in the corpus which are sense-tagged or not. Supervised learning methods have good learning ability and can get better accuracy in word sense disambiguation experiments, Hinrich Schutze [10]. In general, data sparseness is a common problem in supervised learning. This can be overcome by data smoothing which is a time consuming task, Giedre Pajarskeite [3]. Unsupervised word sense disambiguation does not depend on tagged corpus and could realize the training of large real corpus coming from all kinds of field. This method may overcome data sparseness problem to some extent. T Pederson [7] used bigram for word sense disambiguation. Contingency table with size 2×2 was defined with frequency of the two consecutive words appearing together, first word appearing without second word, first and second words coming as first and second words of any bigram. When the size of corpora increases, there, data redundancy is unavoidable. Watanabe N et al. [1] used neural network for sense disambiguation. They applied Up/Down state neurons and morphoelectronic transform for firing (activating) the neurons in the network. By using concept distance, the neurons relevant to the words were fired for making disambiguation.

A. Azzini et al, [2] considered a large tagged datasets for every sense of an ambiguous word and adopted Neuro evolutionary algorithm for disambiguation. Along with text features, maximum and minimum distances are treated as additional features for disambiguation. Michael [5] used eight different entity types for 10 distinct word positions, totally, eighty features per instance to classify the named entities using feed forward neural network with 50 hidden units. He used trigram as one of a feature for it. Massimiliano et al [6] used part of speech, surrounding words, bigrams, trigrams and syntactic information as feature set for hierarchical multiclass perception, from that he achieved 71.8% accuracy in word sense disambiguation. Zhimao Lu et al [8] extracted mutual Information (MI) of the words as input vectors for back-propagation neural network and tested with maximum feature sets varying from ten from left and ten words 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 and also to improve the performance. We present this paper in such way to describe about disambiguation process in section 2, experimental results in section 3 and conclusion in section 4.

2. Disambiguation Process

Disambiguation process is done in five phases: pre-disambiguation process, feature representation, vector formation, network construction and sense identification.

2.1. Pre-Disambiguation Process:

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

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

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.

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

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 et al. [12]. Ambiguous words in the input sentence are isolated using WordNet and restructured into five different vector forms: pre-bigram (T_1), post-bigram (T_2), pre-trigram(T_3), in-trigram(T_4) and post-trigram(T_5).

2.2. Feature Representation:

To make disambiguation, minimum two features (bigram) and maximum three (trigram) are taken. Bigrams are tested in two ways, Pre-bigram and Post-bigram, based on the position of the ambiguous word. In the same way, trigrams are considered as Pre-trigram, in-trigram and Post-trigram. Here, word disambiguation is done with minimum two features (bigram) and maximum three features (trigram), along with ambiguous word. It is very difficult to process text as such for any kind of Natural Language Processing task. To overcome the problem, vectorization is done. To make a vector form of the bigram and trigram features of the ambiguous words, in the input sentence and in the training data, first, PoSs in Brown corpus (by ignoring two PoS type in Brown corpus, NPS and POS) are categorized and framed into 17 groups (table-1) based on their characteristic. Next, weight assignment task is done based on the group. Elements in the groups are mutually exclusive in nature. If one element comes in a place in a sentence, there is least chance for other element from the same group to be in the same location. The distributional characteristics of bigrams are consistent across corpora; a majority of them occur only one time. This leads to sparseness, and to overcome the problem, power divergence family of Goodness and Dice coefficient were used to fit the statistics, T Pederson [7]. To avoid the sparseness, a minimal non-zero value .0001, not in the weight group is assigned.

Table-1: Grouping Features based on the category

PoS Type	Weight	PoS Type	Weight	PoS Type	Weight
CC	0.01	PP, PRP$, PRP, NP, PR	0.07	UH	0.13
DT	0.02	WDT, WP, WP$, WRB	0.08	NN, NNP, NNPS, CD	0.14
EX	0.03	NNS	0.09	VB	0.15
FW	0.04	VBD, VBG, VBN, VBP, VBZ, MD, MD|VB	0.10	JJ	0.16
JJR	0.05	RBR, RBS	0.11	RB	0.17
LS	0.06	IN, PDT, TO, RP	0.12		

2.3. Vector Formation

2.3.1. Input sentence

Ambiguous words are represented along with either previous (post-bigram) or immediate next word (pre-bigram) in bigram. And the trigram is represented in three different combinations, two left words and ambiguous word (post-trigram), one left and one right with ambiguous word in the centre (in-trigram) and two right words with ambiguous word (pre-trigram). To frame the vector form of each word of bigram and trigram, relevant weights are substituted instead of the word. For example, the sentence, 'He is working as a manager in a bank", First, all ambiguous words are isolated (Fig-1) and each of their relevant vectors are framed (table-2).

word position	word	PoS	# of senses
3	work	VB	27
6	manager	NN	2
9	bank	NN	10

Fig-1: Ambiguous Words

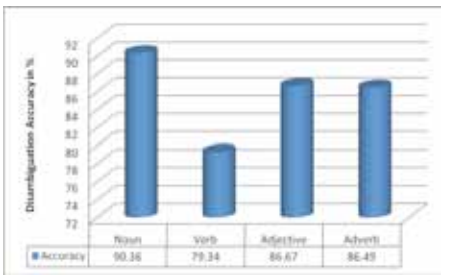

Fig-2: Disambiguation Accuracy Based on Word Catogeroy

Table-2: Framing the vector form for the input Sentence

Features	Ambiguous words		
	work	manager	bank
Pre-Bigram	**.1500** .1200	**.1400** .1200	.0200 **.1400**
Post-Bigram	.1000 **.1500**	.0200 **.1400**	**.1400** .1200
Pre-Trigram	**.1500** .1200 .0200	**.1400** .1200 .0200	**.1400** .0001 .0001
In-Trigram	.1000 **.1500** .1200	.0200 **.1400** .1200	.0200 **.1400** .0001
Post-Trigram	.0700 .1000 **.1500**	.1200 .0200 **.1400**	.1200 .0200 **.1400**

2.3.2. Training data representation (Distributed Electronic Dictionary (DED) Construction):

For a supervised disambiguation, sense tagged training data play in a very important role. XML representations of Brown Corpus data were refined as easy to use form, P.Tamilselvi et al [9]. Ambiguous words in 8000 sentences from refined corpus are treated as the training data and are arranged in a distributed form of Electronic dictionaries (DED). Algorithm used for constructing DED is given in Fig-2.

2.4. Neural Network Architecture:

Three different neural networks, namely, feed-forward back propagation network (M1), trainable cascade forward back propagation network (M2) and pattern recognition network (M3) is taken for process. Almost all networks are designed to have a single hidden layer having neurons in the multiples of five from five to twenty. Tangent Sigmoid transfer function is applied in hidden layer and linear 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 absolute error function ***(mae)***, mean squared error function (***mse)***, mean squared error with regularization performance function ***(msereg)*** and Sum squared error performance function ***(sse)*** are used. The networks are adopted and trained by changing the weights repeatedly for producing better result.

2.5. Sense Disambiguation:

Disambiguation is one of the main tasks for Natural Language Processing. Different ways of techniques such as Statistical approach, Marine Carpuat et al [14], decision tree, Pedersen [7], and artificial neural network, Zhimao Lu et al [8] etc are used for disambiguation. In this paper, three different feed forward back propagation networks are constructed for disambiguation. Neural network architectures are explained in the next section. Disambiguation performances of them are compared to select the best architecture for the process.

DED Construction Algorithm:

i=1;j=0;k=1;
fp=fopen(BC_refinedfile);
while not_eof(fp)
str=get(S_i); [get ith sentence]
j=j+1;
while any(w_{ij}) [if any word 'w_{ij}' in the sentence S_i]
if ambiguous(w_{ij})==true [if 'w_{ij}' is an ambiguous word]
W=weight(PoS(w_{ij}))
Switch (j)
case 1: $l_3=l_2=l_1=0.0001$;
case 2: $l_3=l_2=0.0001$; l_1=weight(PoS(w_{ij-1}));
case 3: l_3= 0.0001; l_2=weight(PoS(w_{ij-2})); l_1=weight(PoS(w_{ij-1}));
case n-2:r_1=weight(PoS(w_{ij+1})); r_2=weight(PoS(w_{ij+2})); r_3=0.0001;
case n-1: r_1=weight(PoS(w_{ij+1})); $r_2=r_3$=0.0001;
case n: $r_1=r_2=r_3$=0.0001;
otherwise : l_1=weight(PoS(w_{ij-1})); l_2=weight(PoS(w_{ij-2})); l_3=weight(PoS(w_{ij-3}));
r_1=weight(PoS(w_{ij+1})); r_2=weight(PoS(w_{ij+2})); r_3=weight(PoS(w_{ij+3}));
end; [switch]
$train_k$=[w_{ij} sense_value sense_tag l_3 l_2 l_1 W r_1 r_2 r_3]
k=k+1;
end; [while any word in the sente]
end; [end of file]
utrain=unique(sort(train)); [sorted unique training data]
[r c]=size(utrain);
for ii=1 to r
switch first_character(utrain(w)$_{ii}$)
case 'A' or 'a': DED_A =write($utrain_{ii}$);
Case 'B' or 'b': DED_B =write($utrain_{ii}$);
:
:
Case 'Z' or 'z': DED_Z =write($utrain_{ii}$);
end; [switch]
end; [for loop]

Fig-2: Algorithm for DED construction

Table-3: Performance Evaluation using different error values

Post-Bigram	M1				M2				M3			
Neurons	mae	mse	msereg	sse	mae	mse	msereg	sse	mae	mse	msereg	sse
5	0.61	0.67	0.59	1002.83	0.91	0.41	0.41	1001.24	0.84	0.59	0.37	2727.84
10	0.97	0.56	0.58	1022.27	0.55	0.42	0.35	1020.64	0.79	0.68	0.62	2028.77
15	0.85	0.58	0.59	1281.23	0.90	0.59	0.48	1094.08	0.65	0.59	0.66	1094.08
20	0.80	0.59	0.55	2402.83	0.65	0.58	0.50	1414.12	0.72	0.54	0.07	1414.12
Pre-Bigram	M1				M2				M3			
Neurons	mae	mse	msereg	sse	mae	mse	msereg	sse	mae	mse	msereg	sse
5	0.73	0.25	0.54	1149.40	0.93	0.67	0.31	989.83	0.70	0.48	0.60	199.80
10	0.63	0.34	0.44	978.41	0.70	0.59	0.29	1018.54	0.86	0.54	0.45	299.66
15	0.70	0.55	0.48	1070.86	0.88	0.54	0.48	216.94	0.80	0.54	0.53	216.94
20	0.83	0.68	0.50	1005.20	0.51	0.54	0.49	1864.54	0.87	0.54	0.09	1864.54
Pre-Trigram	M1				M2				M3			
Neurons	mae	mse	msereg	sse	mae	mse	msereg	sse	mae	mse	msereg	sse
5	1.13	0.67	0.54	65535.00	0.81	0.71	0.48	65535.00	0.81	0.94	0.99	65535.00
10	0.75	0.61	0.53	65535.00	0.69	0.64	0.47	65535.00	0.81	0.94	0.99	65535.00
15	0.76	0.38	0.33	65535.00	1.01	0.63	0.54	65535.00	0.81	0.94	0.99	65535.00
20	1.03	0.99	0.97	65535.00	0.81	0.95	0.99	65535.00	0.81	0.94	0.99	65535.00
In-Trigram	M1				M2				M3			
Neurons	mae	mse	msereg	sse	mae	mse	msereg	sse	mae	mse	msereg	sse
5	0.61	0.66	0.19	818.97	0.99	0.55	0.50	962.13	0.87	0.66	0.42	361.58
10	0.87	0.64	0.57	989.57	0.76	0.63	0.31	974.47	0.78	0.54	0.49	335.64
15	0.70	0.54	0.48	1060.10	0.64	0.55	0.50	319.74	0.71	0.63	0.68	319.74
20	0.85	0.51	0.44	947.74	0.60	0.50	0.44	937.37	*0.59*	*0.58*	*0.06*	*937.37*
Post-Trigram	M1				M2				M3			
Neurons	mae	mse	msereg	sse	mae	mse	msereg	sse	mae	mse	msereg	sse
5	0.89	0.20	0.42	980.41	0.74	0.39	0.42	980.41	0.91	0.49	0.32	211.50
10	1.04	0.30	0.56	1162.60	0.88	0.51	0.54	1063.51	0.65	0.47	0.44	941.05
15	0.76	0.53	0.48	989.49	0.75	0.49	0.29	462.35	0.67	0.61	0.58	462.35
20	0.58	0.22	0.22	1441.07	0.78	0.69	0.63	230.68	0.95	0.58	0.09	230.68

3. Experimental Results:

More than 500 sentences from Brown Corpus which are not a part of the training data are taken and tested with the networks. The disambiguation task is processed with three different network architectures, M1, M2 and M3. Hidden neurons are changed from 5 to 20 with the increase of 5 neurons in a step. Size of input neurons for post-bigram and pre-bigram are two and three for pre-trigram, in-trigram and post-trigram. The performance is measured by four different error functions, ***'mae', 'mse', 'msereg' and 'sse'.*** Disambiguation accuracy is high when the error difference is less. It is explicitly realized from table-3 (different error values) and table-4 (disambiguation accuracy in %). Error values in M3 neural network with 20 hidden neurons with in-trigram features are comparatively less. For the same architecture, the disambiguation accuracy reaches 85.72%. Disambiguation is also measured based on the PoS categories. In general, noun, verb, adjective and adverb will fall under ambiguous category. Disambiguation accuracy is measured based on these four PoS categories from pattern recognition network (M3) with 20 neurons with In-trigram features, given in fig.3. From the chart, nouns are identified correctly with accuracy of 90.36% but verbs are not up to that accuracy level.

Table-4: Disambiguation Accuracy in %

Feature Type	% of Accuracy based on Number of Neurons in the hidden layer				Maximum Accuracy Level
Post-Bigram	5N	10N	15N	20N	
M1	64.29	57.15	71.43	64.29	71.43
M2	64.29	64.29	64.29	71.43	71.43
M3	78.58	64.29	71.43	78.58	78.58
Pre-Bigram	5N	10N	15N	20N	Maximum
M1	71.43	71.43	71.43	71.43	71.43
M2	78.58	71.43	71.43	71.43	78.58
M3	78.58	78.58	78.58	78.58	78.58
Pre-Trigram	5N	10N	15N	20N	Maximum
M1	64.29	64.29	71.43	71.43	71.43
M2	64.29	71.43	71.43	71.43	71.43
M3	71.43	71.43	71.43	71.43	71.43
In-Trigram	5N	10N	15N	20N	Maximum
M1	64.29	57.15	78.58	71.43	78.58
M2	64.29	71.43	57.15	78.58	78.58
M3	64.29	71.43	71.43	**85.72**	**85.72**
Post-Trigram	5N	10N	15N	20N	Maximum
M1	64.29	57.15	57.15	78.58	78.58
M2	64.29	64.29	64.29	64.29	64.29
M3	71.43	64.29	64.29	64.29	71.43

4. Conclusion

In this paper, disambiguation problem is tried with minimal feature set. Disambiguation accuracy was achieved as 71.8% using hierarchical multiclass perceptron with gargantuan features, Massimiliano Ciaramita et al [6]. Andy Zuppann [11] used F-Measure and derived the accuracy as 82.6%, with all surrounding words without encoding the word relations. Here, word ambiguity was solved by three different neural networks with one hidden layer having 5 to 20 neurons by increasing them in the multiples of 5 with maximum three features. We obtained 85.72% of disambiguation accuracy from pattern recognition neural network with 20 hidden neurons using in-trigram feature set. Accuracy level based on the PoS category is also measured. Verb disambiguation accuracy, 79.34%, is the lowest accuracy when compared to other categories. We achieved better performance with very few features.

References:

[1]. Watanable N and Ishizaki S, "Neural Network Model for Word Sense Disambiguation using Up/Down state and Morphoelectrotonic Transformation" , *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 2007, 11(7), p. 780 – 786.

[2]. Azzini and C. da Costa Pereira and M. Dragoni and A. G. B. Tettamanzi, "Evolving Neural word sense disambiguation classifier with a letter-count distributed encoding" ,*WIVACE08 – Proceedings*, 2008.

[3]. Giedre Pajarskeite, Vilma Griciute and Gailius Raskinis, "Designing HMM-based part-of-speech tagger for Lithuanian language, Informatica", 2004, 15(2), p. 231-242.

[4]. Ng. H.T. and Zelle J, "Corpus-Based Approaches to Semantic Interpretation in Natural Language Processing", AI Magazine, 1997, 18(4), PP. 45-64.

[5]. Michael Fleischman and Eduard Hovy, "Fine grained classification of named entities", In *Proceedings of ACL-2002*, Morristown, NJ, USA, 2002, p. 1-7.

[6]. Massimiliano Ciaramita, Thomas Hofmann, and Mark Johnson, "Hierarchical semantic classification: Word sense disambiguation with world knowledge", in the *18th International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, 2003.

[7]. T. Pedersen, "A decision tree of bigrams is an accurate predictor of word senses", Second Annual Meeting of the North American Chapter of the Association for Computational Linguistics, 2001.

[8]. Zhimao Lu, Ting Liu, and Sheng Li. "Combining neural networks and statistics for chinese word sense disambiguation", in Oliver Streiter and Qin Lu *Workshop,* 2004*,* p. 49-56,.

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

[10]. Hinrich Schutze, Automatic word sense discrimination, *Computation Linguistics*, 1998, 24(1), pp.97-124.

[11]. Andy Zuppann, A, Conncectionist Approach to word sense disambiguation, Proceeding of the conference Class of 2003 Senior Conference on Natural Language Processing, Swarthmore College, USA, 2003.

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

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

[14]. Marine Carpuat and Dekai Wu, "Improving Statistical Machine Translation using Word Sense Disambiguation", *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 61–72.