A Survey of Sense Annotation, Eye-tracking and Wordnet Linking approaches till June, 2012

A Literature Survey
Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology

Arindam Chatterjee
Roll No: 09305905

Under the guidance of
Prof. Pushpak Bhattacharyya

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay

June 30, 2012

Abstract

Here we present a survey of important work done on Sense Annotation or Word Sense Disambiguation (WSD) between the period 1985 to 2012. We also present work done on eye-tracking between 2008 to 2012 and Wordnet Linking work done between 2000 to 2012. Word Sense Disambiguation (WSD) is defined as the task of computationally finding the senses of words from a *context*. Eye-tracking relates to tracking eye fixations on certain units pre-defined for an experiment. It is a psycholinguistic approach for studying eye movements and hence cognition with respect to various Natural Language Processing (NLP) tasks. Wordnet linking is the task of linking the senses of Wordnets of two or more languages. It requires both man and machine sense determination techniques.

Contents

Chapter 1

Sense Annotation

In this chapter, we provide a brief overview of the existing work on WSD. To keep it in line with the problems addressed in this thesis, we will have a look at the Supervised, Semi-Supervised and Hybrid approaches to WSD, both monolingual, as well as bilingual.

1.1 Supervised Algorithms

In the last fifteen (15) years, the NLP community has witnessed an increasing interest in machine learning based approaches for automated classification of word senses. This is evident from the number of supervised WSD approaches that have spawned. Today, the supervised approaches for WSD possibly are the largest number of algorithms, used for disambiguation. Supervised WSD uses machine learning techniques on a sense-annotated data set to classify the senses of the words. There are a number of classifiers also called word experts that assign or classify an appropriate sense to an instance of a single word. The training set for these algorithms consist of a set of examples, where the target word is manually tagged with sense from a reference dictionary. The supervised algorithms thus perform target-word WSD. Each algorithm uses certain features associated with a sense for training. This very fact forms the common thread of functionality of supervised algorithms. In this section we will discuss the major supervised algorithms for sense disambiguation in the literature.

1.1.1 Decision Lists

The decision list is a set of rules in an ordered list format. It is a set of weighted "*if-then-else*" rules. It was first used by (Yarowsky, 1995) on the `SENSEVAL` corpus. It is one of the most efficient supervised algorithms. First, the features are extracted from the set of training examples, which in this case is the training corpus. This is followed by the testing phase, where the WSD algorithm is run. This is based on a probabilistic measure.

1.1.1.1 Feature Extraction

The feature extraction phase is the training phase of this algorithm. The features are extracted and stored in a table in an ordered list format. A sense-tagged corpus is taken as a knowledge source. The feature vector for each word *w* has the following features in it:

- *Part-Of-Speech(POS)* of *w*
- *Semantic & Syntactic feature*s of *w*
- *Collocation vector* (set of words around it) - typically consists of next word(+1), next-to-next word(+2), -2, -1 & their POS's.
- *Co-occurrence vector* - number of times *w* occurs in bag of words around it.

The method is based on a heuristic:
'**One sense per collocation**' property - Nearby words provide strong and consistent clues as to the sense of a target word.

1.1.1.2 Generation of Decision Lists

Once the features are obtained from the corpus, rules of the form, (feature value, sense, score) are created. These rules are embedded into a table, one entry for each sense. This table is then sorted in decreasing order of scores. The resultant data structure, *i.e.*, the sorted table is the decision list. The next question that arises is how to calculate a score for a sense, given its features. Each sense has a feature vector comprising of a number of features, as shown earlier. The task is to find the feature in the feature vector, which contributes most to the appropriateness of the sense. For this, the score of the features needs to be calculated and the maximum feature score can be used as the sense score. According to Yarowsky, the score of a feature f is computed as the logarithm the probability of sense S_i given feature f divided by the sum of the probabilities of the other senses given feature f :

$$score(S_i) = max_f log\left(\frac{P(S_i|f)}{\sum_{j\neq i} P(S_j|f)}\right)$$

The above formula is that of (Agirre and Martinez, 2000), which is an adaptation of the two sense formula of Yarowsky. The probabilities $P(S_j|f)$ can be estimated using the maximum-likelihood estimate. Smoothing and Pruning can be used to avoid zero counts and to eliminate unreliable rules with very low weight.

Given a word *w* to be disambiguated along with its feature vector, the decision list is scanned for the entries that match the input vector. The sense with the maximum score among the entries becomes the winner sense.

1.1.2 Decision Trees

The decision tree (Rivest, 1989) is a prediction based model. The knowledge source used for the decision tree is a sense-tagged corpus, on which the training is done. The classification rules in case of decision tree are in the form of "*yes-no*" rules. Using these rules the training data set is recursively partitioned. The decision has the following characteristics:

- Each internal node represents a feature, on which a test is conducted.
- Each branch represents a feature value, or an outcome of the test on the feature in the internal node.

- Each leaf node represents a sense or a class.

The feature vector used in the case of decision tree, is the same as that of decision list. The feature vector for each word *w* has the following features in it:

- *Part-Of-Speech(POS)* of *w*
- *Semantic & Syntactic feature*s of *w*
- *Collocation vector* (set of words around it) - typically consists of next word (+1), next-to-next word(+2), -2, -1 & their POS's.
- *Co-occurrence vector* - number of times *w* occurs in bag of words around it.

1.1.2.1 Generation of Decision Tree

Once the features of the sense are in place, the decision tree is generated using **ID3, ID4, ID5** or, **ID5R** algorithms. The basic one among these algorithms is the **ID3** algorithm, which is similar to the **C4.5** algorithm *Quinlan* (Quinlan, 1986) . The **ID3** algorithm can be stated as follows:

- If all the instances are from exactly one class, create a leaf node containing that class name.
- Else, for each node, find the feature with least Entropy value and grow the sub-trees recursively using values of that attribute.

1.1.2.2 The WSD algorithm

Once a word w is up for disambiguation, along with its feature vector, using the already gathered training information, the decision tree is traversed to reach a **leaf** node. The sense contained in the leaf node gives the *winner sense*.

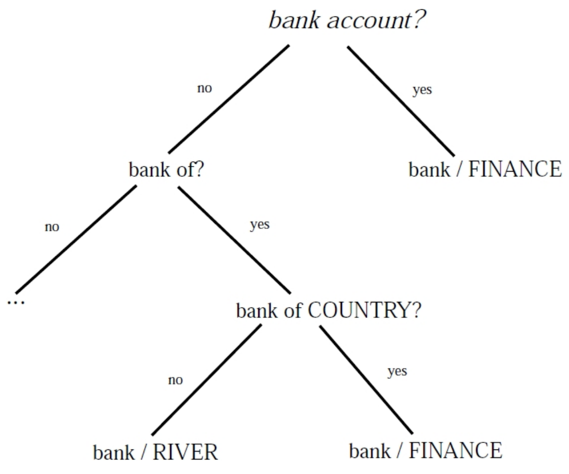


Figure 1.1: An Example of Decision Tree

1.1.3 Neural Networks

A Neural Network *McCulloch and Pitts* [McCulloch and Pitts, 1943] is an interconnection of artificial neurons, used for classification of patterns(data), based on a connectionist approach. There are many kinds of neural networks, like perceptrons, feed-forward, recurrent networks. The neural networks used for WSD purpose are: Perceptrons using `Hidden Markov Mode` (HMM) and `Back propagation` based feed forward networks. In case of WSD using Perceptron trained HMM, the WSD problem is treated as a sequence labeling task. The class space is reduced by using super senses instead of actual senses from the WordNet. The HMM is trained using the following features:

- **POS** of *w* .
- **POS of neighboring** words.
- Local **collocations**.
- **Shape** of the word and neighboring words.

Example:
For s = "*Merrill Lynch & Co shape*(s) = $Xx*Xx*\&Xx$

This method is useful for Named entity recognition, as labels like *"person"*, *"location"*, *"time" etc.*. are included in the super sense tag set. The other type of neural network that is used for WSD purpose is the feed-forward network. This network consists of three layers of neurons, namely Input layer, Hidden layer and Output layer. The feedforward network, trains by learning the weights of the connections and the threshold values of the hidden layer and output layer neurons. It takes the feature vector as input. The number of input layer neurons, depends on the size of the feature vector, *i.e.* one input neuron for each feature. The inputs though are binary. During testing, given a target word *w*, and its set of features, the inputs for the features present in the feature vector are set to 1, rest to 0. Correspondingly a neuron in the output layer fires. Each output layer neuron corresponds to a sense of *w*. The sense associated with the neuron that fired becomes the winner sense.

1.1.4 Exemplar/Memory Based Learning

Exemplar based (or instance based or memory based) learning [Navigli and Velardi, 2005a] is based on learning from examples. The model stores the examples as points in the feature space. It is called memory based, because as new examples are added, new models are not created, rather they are progressively added to the existing model.

The most commonly used method for this approach is the *k-Nearest Neighbor*(kNN) method. It is one of the best performing methods in WSD.

In *k*NN method, a new example is classified based on the *k* most similar examples that were stored earlier. Formally, a new example say $x = (w1, w2, ..., wn)$ which is expressed in terms of *m* features is classified by the closest *k* neighbors. The closeness is mathematically computed by the distance, for example the *Hamming distance*:

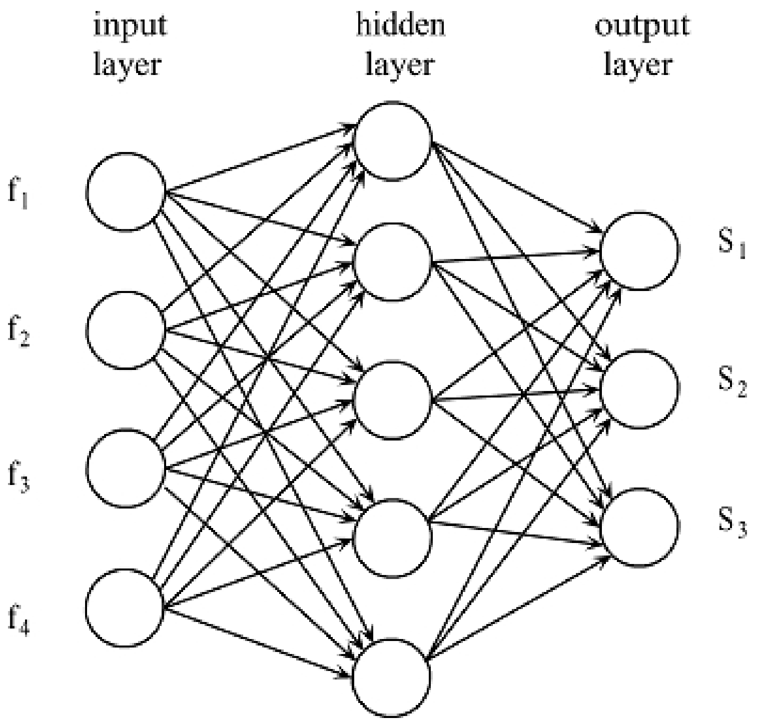


Figure 1.2: An example of feed-forward network for WSD

$$\delta(x_i,x_j) = \Sigma_{j=1}^{m}\partial(x_i,x_j)$$

where:
w_j : weight of the jth feature.
$x_i = (x_{i_1}, x_{i_2}, ..., x_{i_m})$: a previously stored example.
$\partial(x_i,x_j) = 0$ if $x_i = x_i$ and = 1 otherwise.

The set of k closest instances is derived to form a set say $Closest_k$. The new example x belongs to that class (sense) which has the largest number of members in $Closest_k$, *i.e.*, x belongs to that class that has the highest number of neighbors of x.

1.1.4.1 Determining the weights w_j and the value of k

The value of k is determined experimentally. Feature weights w_j can be estimated, for example,with the *gain ratio measure* (Quinlan, 1996). More complex metrics, like the *Modifed Value Difference Metric* (MVDM) (Cost and Salzberg, 1993), can be used to calculate graded distances between feature values, but usually they are computationally more expensive.

1.1.5 Ensemble Methods

Since a lot of work has gone into supervised approaches for WSD, and there are a lot of supervised algorithms for sense disambiguation today, a combination of such strategies could result in a highly efficient supervised approach and improve the overall accuracy of the WSD process. Features should actually be chosen so that significantly different,

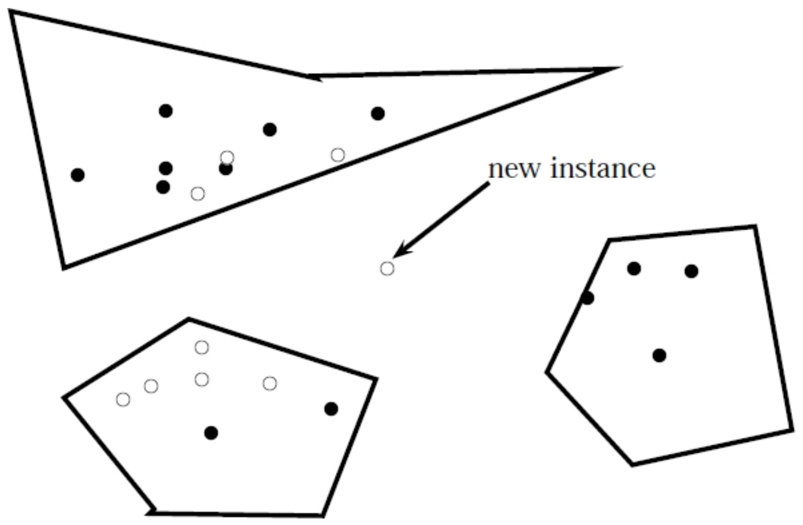


Figure 1.3: An example of kNN on 2D plane

possibly independent, views of the training data (*e.g.*, lexical, grammatical, semantic features, *etc.*) are formed. These combination strategies are called ensemble methods. One of the cheif ensemble methods is majority voting, described below. The ensemble strategy that has highest accuracy is the AdaBoost method.

1.1.5.1 Majority Voting

In the majority voting scheme, each classifier votes for a particular sense of the given word w. A classifier votes for a sense S_i of the word w, if that sense is the output, or the winner sense for that classifier. The sense with the majority of votes becomes the winner sense for this method. Formally, given w, the senses of w S_i and the ensemble components C_j. The winner sense $\hat{S}$ is found out by the formula:

$$\hat{S} = argmax_{S_i \in Senses_D(w)} |j : vote(C_j) = S_i|$$

If there is a tie, then a random choice is made among the winner senses or the ensemble does not output anything.

1.1.5.2 AdaBoost

Adaboost is a theoretical framework of a machine learning model called `Probably Approximately Correct` (PAC). The method is sensitive to noisy data and outliers, and is consequently less susceptible to overfitting than other machine learning approaches. AdaBoost or *Adaptive Boosting* (Freund et al., 1999) constructs a *"strong"* classifier by taking a linear combination of a number of *"weak"* classifiers. The method is called Adaptive because it tunes classifiers to correctly classify instances misclassified by previous classifiers.

For learning purposes, instances in the training data set are equally weighted initially. AdaBoost learns from this weighted training data set. For m ensemble components,it iterates m times, one iteration for each classifier. In each iteration, the weights of the misclassified instances are increased, thus reducing the `overall classification error`.

As a result of this method, after each iteration $j = 1, ..., m$ a weight α_j is obtained for each classifier C_j, which is a function of the classification error for C_j, over the training set. Given the classifiers $C_1, C_2, ..., C_m$ the attempt is to improve α_j which is the weight or importance of each classifier. The resultant "strong" classifier H can thus be formulated as:

$$H(x) = sign(\Sigma_{j=1}^{m} \alpha_j C_j(x))$$

This indicated that H is the sign function of a linear combination of the "weak " classifiers. An improved version of AdaBoost called *AdaBoost.MH* (Schapire and Singer, 1999) is also available in the literature. An application of AdaBoost called *LazyBoosting* was also used by *et al.* (Escudero et al., 2001). LazyBoosting is basically AdaBoost used for WSD purpose.

1.1.6 Support Vector Machines

Support Vector Machines were introduced by (Boser et al., 1992) is based on the idea of learning a *hyperplane*, from a set of the training data. The hyperplane separates positive and negative examples. The hyperplane is located in the hyperspace, such that it maximizes the distance between the closest positive and negative examples (called *support vectors*). The SVM thus minimizes the `classification error` and maximizes the geometric distance or margin between the positive and negative examples. The linear SVM is characterized by two parameters:

- ***w***, which is the vector perpendicular to the hyperplane.

- ***b***, the bias which is the offset of the hyperplane from the origin.

An instance is labeled as positive if the value $f(x) = w.x + b \geq 0$ and negative otherwise. The diagram given above, shows the support vectors and the separating hyperplane along with w and b. This can thus be well understood from the geometric intuition as shown above.

SVM is a binary classifier, but WSD is a multiclass problem, as there can be more then two senses(classes) for a word. To make it usable for WSD, the problem can be broken down into a number of binary class problems.

This can be done by taking each sense as one class and the remaining senses as another class. This is done for all the senses. The sense with the maximum confidence score is taken as the winner sense. The confidence score is actually the value of **f(x)**$[w.x + b]$, for each SVM.

1.1.7 SNoW Architecture

SNoW stands for ***Sparse Network Of Winnows***, which is basically an online learning algorithm. The fundamental construct of the algorithm is the *Winnow algorithm* [Escudero et al., 2000]. The algorithm learns very fast in the presence of many binary

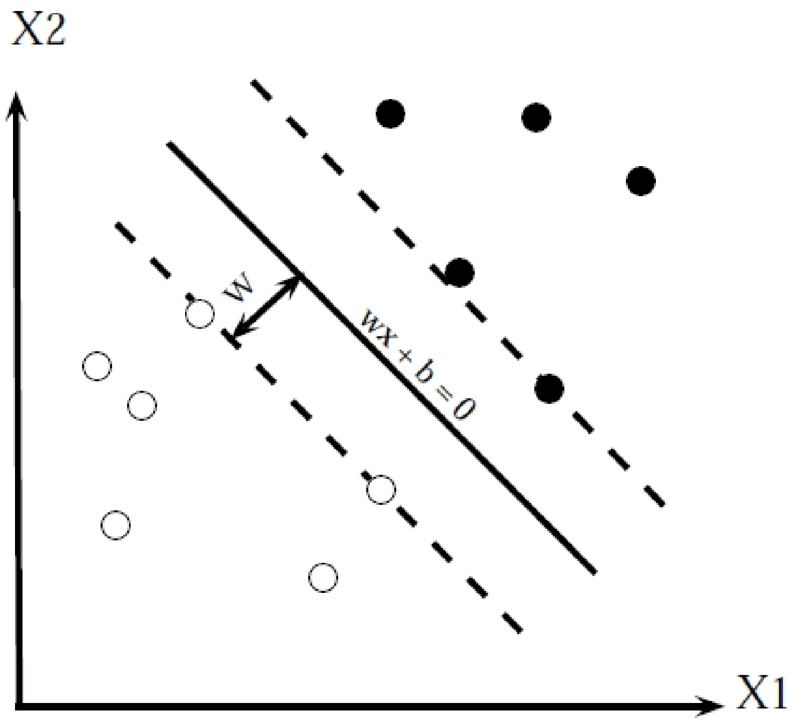


Figure 1.4: The geometric intuition of SVM

input features, as it consists of a linear threshold algorithm and updates multiplicative weight for problems having 2 classes.

Each class in the SNoW architecture has a winnow node, which learns to separate that class from the remaining classes. During training, if an example belongs to the corresponding class, then it is considered positive for the winnow node, else it is a negative example. The nodes are not connected to all features; rather they are connected to *"relevant"* features for their class only. This accounts for the fast learning rate of SNoW.

When classifying a new example, SnoWbehaves somewhat like a neural net, which takes features as input and outputs the class with the highest activation value. According to [Zwirello, 2005], SNoW performs well in higher dimensional domains. Both the target function and the training instances are sparsely distributed in the feature space, *e.g.*: `text categorization, context sensitive spelling correction, WSD`, *etc.*

1.2 Semi-supervised Algorithms

Supervised algorithms train a model based on the annotated corpus provided to it. This corpus needs to be manually annotated, and the size of the corpus needs to be large enough in order to train a generalized model.

Semi-supervised, also known as *minimally* supervised algorithms make some assumptions about the language and discourse in order to minimize these restrictions. The **common thread** of operation of these algorithms are these `assumptions` and the `seeds` used by them for disambiguation purposes.

This section presents two such approaches, based on two different ways to look at the problem, namely Bootstrapping and Monosemous Relatives.

1.2.1 Bootstrapping

This algorithm, devised by Yarowsky [Yarowsky, 2000], is based on Yarowsky's supervised algorithm that uses Decision Lists. As mentioned earlier, the algorithm makes a couple of assumptions regarding the language. The assumptions can be stated as follows:

- **One sense per Collocation** - The sense of a word is strongly dependent on the neighboring words.
- **One sense per Discourse** - Every document contains a single sense of a word with high proba- bility.

It can be seen that these assumptions are very strong, and thus the model building phase becomes quite small compared to the supervised analogue of this algorithm. With these assumptions, the algorithm first identifies a set of seed words, which can act as disam- biguating words. A Decision List is built based on this seed data. Next, the entire sample set is classified using the Decision list generated previously. Using this decision list, as many new words as possible are classified in order to identify their senses. Using these words along with their identified senses, new seed data is generated. The same steps are repeated until the output converges upto a threshold value.

1.2.2 Monosemous Relatives

With exponential growth of the *world wide web*, approaches are being tried out which can use the vast col- lection of words as corpus. This enables the algorithms to have an automatically annoatated corpus, which has tremendously huge size, the *web corpus*.

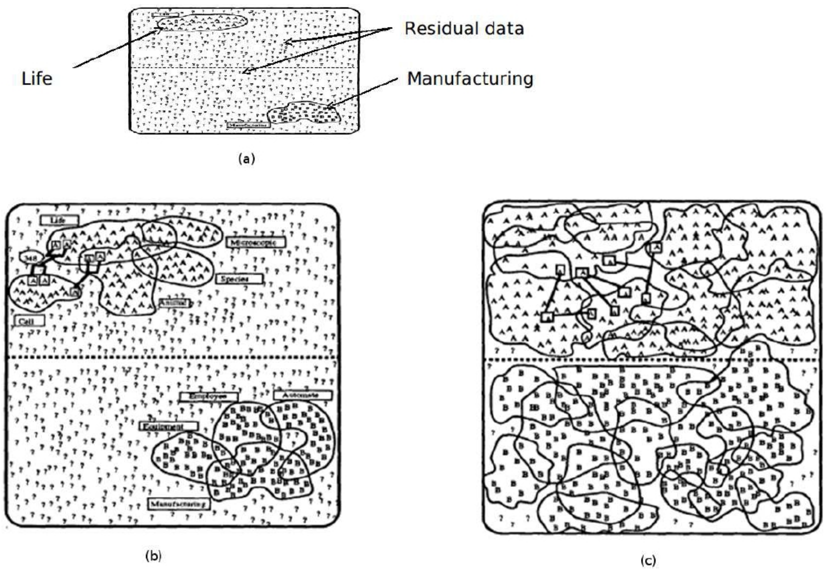


Figure 1.5: figure showing growth of Semi-supervised decision list on two senses of plant *viz.* life and manufacturing. (a) The initial seed data. (b) Growth of the seed set. (c) Seed data converges.

Monosemous relatives approach [Gonzalo et al., 2003] is developed as a bootstrapping algorithm to use words with single sense as possible synonyms. For this, through the synset of a word *w*, all words having single sense (the sense of *w* itself) are found. For each word s ∈ this set, a web search is done and contexts are found. These contexts are directly sense annotated with sense of word *w*. A small variant here is to create *topic signatures* containing closely related words associated with each word sense. A manual inspection is necesary for such approaches.

1.3 Hybrid Algorithms

This chapter presents notable algorithms which derive their power from using a lexical resource to obtain relationships and at the same time, use a minimally annotated corpus to build a language model. In case of Hybrid approaches, the common thread of functioning is the use of both, a lexical resource and a minimally annotated corpora. The three algorithms discussed are SenseLearner, Structural Semantic Interconnections(SSI) and Iterative WSD.

1.3.1 SenseLearner

A minimally supervised algorithm was proposed by Mihalcea and Csomai [Rada, 2005], which uses a relatively small corpus as training set and derives the rest of the information from WordNet. The basic steps for the algorithm are:

- From the test corpus, find words which have at least one occurance in the corpus
- Define a semantic model for one or more word categories (*e.g.*, Noun, Verb *etc.*)
- Build (train) the model and from the test corpus make note of words which do not have any occurance in training corpus.
- Assign the first sense to these words from the WordNet.

The use of WordNet allows the algorithm to generalize the model and get a better version of Lin's approach using these semantic dependencies. *e.g.*: An occurance of structure like *'eat rice'* enables to algorithm to use WordNet hypernymy relationship to derive a generalized model like *'consume food'*. Subsequent occurances of similar structures from the test corpus, like *'eat pineapple'* can directly be disambiguated using this generalized model.

1.3.2 Structural Semantic Interconnections

This method developed by [Navigli and Velardi, 2005b] is inspired from lexical chains, and makes use of an extensive lexical resource, the WordNet. It uses the following semantic relations in the WordNet:

- **hypernymy** (car is a kind of vehicle) denoted by (*kind-of*)
- **meronymy** (room has-part wall) denoted by (*has-part*)
- **holonymy** (the inverse of meronymy) denoted by (part-of)

- **pertainymy** (dental pertains-to tooth) denoted by (*pert*)
- **attribute** (dry value-of wetness) denoted by (*attr*)
- **similarity** (beautiful1 similar-to pretty) denoted by (*sim*)
- **gloss** denoted by (*gloss*)
- **context** denoted by (*context*)
- **domain** denoted by (*dl*)

It uses monosemic words as a seed. It also uses collocation information, which represents semantic relatedness between a pair of senses. The collocations are extracted from resources, such as the Oxford Collocations, the Longman Language Activator, and collocation web site. As shown above, each sense is mapped to a WordNet sense and transformed to a relatedness edge.

1.3.2.1 Creation of SSI graph

Given a local word context $C = (w1, w2, ..., wn)$, SSI constructs a graph $G = (V, E)$, where V represents senses of all the words in context C, one vertex for each sense and $E = (s, s')$. This means if there is an interconnection j between s (a sense of the target word) and s' (a sense of its context), in the knowledge base, an edge is drawn. Valid interconnections are created beforehand, manually by a small Context Free Grammar.

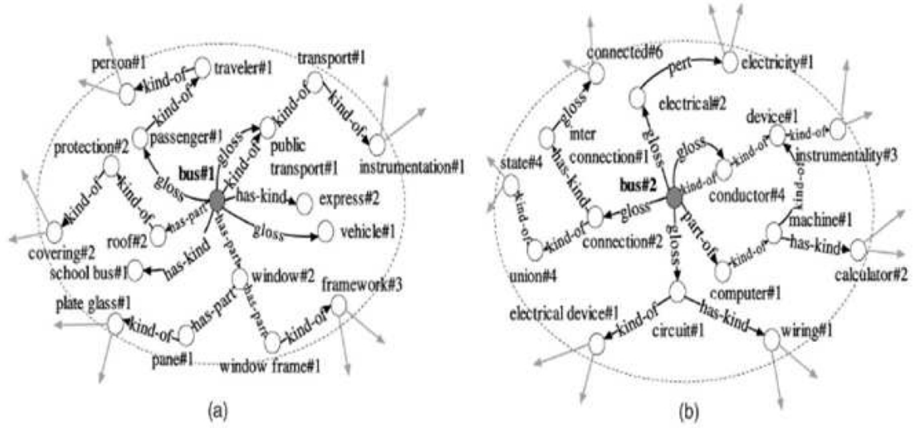


Figure 1.6: A semantic relations graph for the two senses of the word bus (*i.e.*, vehicle and connector)

1.3.2.2 WSD Algorithm

The WSD algorithm is iterative in nature. At each step, for each sense s of a word in C (the set of senses of words yet to be disambiguated), the algorithm determines the degree of connectivity between s and the other senses in C:

Where, *Interconn(s,s')* is the set of interconnections between senses s and s. The contribution of a single interconnection is determined by the reciprocal of its length, which is equal to the number of edges between its ends. The overall degree of connectivity is then normalized by the number of contributing interconnections. The highest ranking sense s of word w_i is chosen and the senses of w_i are removed from the context C. The procedure terminates when either C is empty or there is no sense such that its SSI Score exceeds a particular threshold.

Chapter 2

Bilingual WSD

The failure of monolingual approaches to deliver high accuracies for all-words WSD at low costs created interest in bilingual approaches which aim at reducing the annotation effort. Here again, the approaches can be classified into two categories, *viz.*, (i) approaches using parallel corpora and (ii) and approaches not using parallel corpora.

The approaches which use parallel corpora rely on the paradigm of *Disambiguation by Translation*. Such algorithms rely on the frequently made observation that a word in a given source language tends to have different translations in a target language depending on its sense. Given a sentence-and-word-aligned parallel corpus, these different translations in the target language can serve as automatically acquired sense labels for the source word. Although these algorithms [Diab and Resnik, 2002, Ng et al., 2003] give high accuracies, the requirement of a significant amount of bilingual parallel corpora may be an unreasonable demand for many language pairs (perhaps more unreasonable than collecting sense annotated corpora itself). Further, these algorithms have been tested on a limited set of target words or on a particular POS category (mainly nouns) and may not scale well in an all words scenario.

The second kind of approaches do not use parallel corpora and just rely on in-domain corpora from two languages. For example, [Li and Li, 2004] proposed a bilingual bootstrapping approach for the more specific task of Word Translation Disambiguation (WTD) as opposed to the more general task of WSD. This approach does not need parallel corpora (just like our approach) and relies only on in-domain corpora from two languages. However, their work was evaluated only on a handful of target words (9 nouns) for WTD as opposed to the broader task of WSD. Our work instead focuses on improving the performance of all words WSD.

Bilingual approaches have also been tried for several other NLP taks such as Part-of-Speech Tagging [Yarowsky and Ngai, 2001], Named Entity Recognition [Yarowsky and Ngai, 2001], Chunking [Yarowsky and Ngai, 2001], Sentiment Analysis [Mihalcea and Banea, 2007] and more recently for Semantic Role Labeling [Mukund et al., 2010]. All these methods either project annotations from one language to another or project the statistics learned from the annotated corpus of one language to another language. Our approach also uses parameter projection wherein the various parameters learned from the corpus and wordnet of one language are projected to another language.

Chapter 3

Eye Tracking

We used the eye-tracking device to ascertain the fact that contextual evidence is the prime parameter for human sense annotation as quoted by (Chatterjee et al., 2012) who used different annotation scenarios to compare human and machine annotation processes. An eye movement experiment was conducted by (Vainio et al., 2009) to examine effects of local lexical predictability on fixation durations and fixation locations during sentence reading. Their study indicates that local lexical predictability influences in decisions but not where the initial fixation lands in a word. In another work based on word grouping hypothesis and eye movements during reading by (Drieghe et al., 2008), the distribution of landing positions and durations of first fixations in a region containing a noun preceded by either an article or a high-frequency three-letter word were compared. In our current work we use eye-tracking as a tool to make findings regarding the cognitive processes connected to the human sense disambiguation procedure, and to gain a better understanding of "contextual evidence" which is of paramount importance for human annotation. Unfortunately, our work seems to be a first of its kind, as to the best of our knowledge we do not know of any such work done before in the literature.

Chapter 4

Wordnet Linking

For Wordnet Linking the following literature was surveyed.

4.1 Korean WordNet

Several Word Sense Disambiguation based methods have been used to (semi-)automate the linkage of the Korean WordNet with the English WordNet [Lee et al., 2000]. These methods try to exploit the linguistic phenomena and assign scores to the synsets which assist in mapping.

4.1.1 Candidate Synsets

Given a Korean word, using a bilingual dictionary, all the translations are found in the target language. All the synsets in the target language containing these translations are considered as candidate synsets [Lee et al., 2000]. This method is similar to the first strategy of the Assign Procedure adopted by the MultiWordNet.

4.1.2 Maximum Similarity Heuristic

This heuristic is based on the assumption that all translations in English for the same Korean word sense are semantically similar. Hence it tries to find that synset from the candidate synsets which is most similar to rest of them [Lee et al., 2000]. The similarity between two synsets is calculated as:

$$sim(s_1, s_2) = \frac{2 \times level(MSCA(s_1, s_2))}{level(s_1) + level(s_2)}$$

where, *level(s)* is the depth of concept *s* from the root node of the WordNet and *MSCA* is the most specific common ancestor of two synsets. After calculating the similarity scores, the candidate synset having maximum similarity is chosen as the correct mapping.

4.1.3 Prior Probability

This heuristic provides prior probability to each sense of a single translation as score. Hence translations which are monosemous are given the highest score. The score given is inversely proportional to the degree of polysemy of the translation [Lee et al., 2000].

4.1.4 IS-A Relation

This heuristic is based on the assumption that if two Korean words have an IS-A relation, then their translations in English should also have an IS-A relation [Lee et al., 2000].

4.1.5 Word Match

This heuristic is based on the assumption that concepts which are related are expressed using similar content words. Hence it tries to find the total number of shared words between definitions of Korean words given in the dictionary and the gloss part of the corresponding candidate English synsets from the English WordNet [Lee et al., 2000]. The similarity is calculated as:

$$sim(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

where, X is the set of content words from the English definition given in the bilingual dictionary and Y is the gloss of the candidate synset in the English WordNet.

4.2 PanLexicon Project

PanLexicon is a fully automatic and scalable tool to build word multilingual translation lexicons [Sammer and Soderland, 2007]. Each lexicon entry is a *translation set*, a set of words across multiple languages that all share the same implicit word sense, along with illustrative contexts for each word from [Sammer and Soderland, 2007]). It makes use of bilingual machine readable dictionaries, unaligned corpora for each language and corresponding morphological tools [Sammer and Soderland, 2007]. The system has three main stages: *preprocessing stage*, *finding the best matching contexts* and *merging*.

4.2.1 Preprocessing

For each language, corpora are built by indexing each sentence as a separate document. The index stores each sentence as a string of space separated tokens. The sentences are preprocessed (words are lower cased and morphological analysis is done on them). A table of co-occurrence counts between the tokens/words is constructed where two tokens/words co-occur if they appear in the same sentence together. Then the system collects and stores the contexts for each word [Sammer and Soderland, 2007].

4.2.2 Finding the Best Matching Contexts

To find the context of a translation that best matches the context c of a word, the system queries the context index constructed during preprocessing stage with the bag of words consisting of all of the translations of all the tokens from c. The top n contexts returned by the index are then ranked using a scoring metric based on *pointwise mutual information* (PMI). Using this the matching context words of the word are obtained [Sammer and Soderland, 2007].

4.2.3 Merging

The translation sets of all potential synonyms of a given word are collected. The set of potential synonyms of the source word (English) is defined as the intersection of the sets of back translations though each of the other languages. Depending on the cardinality of the set obtained after intersection, decision is made on merging the two sets. This fulfills the goal of translation set merging, which is to combine translation sets representing the same sense [Sammer and Soderland, 2007].

Chapter 5

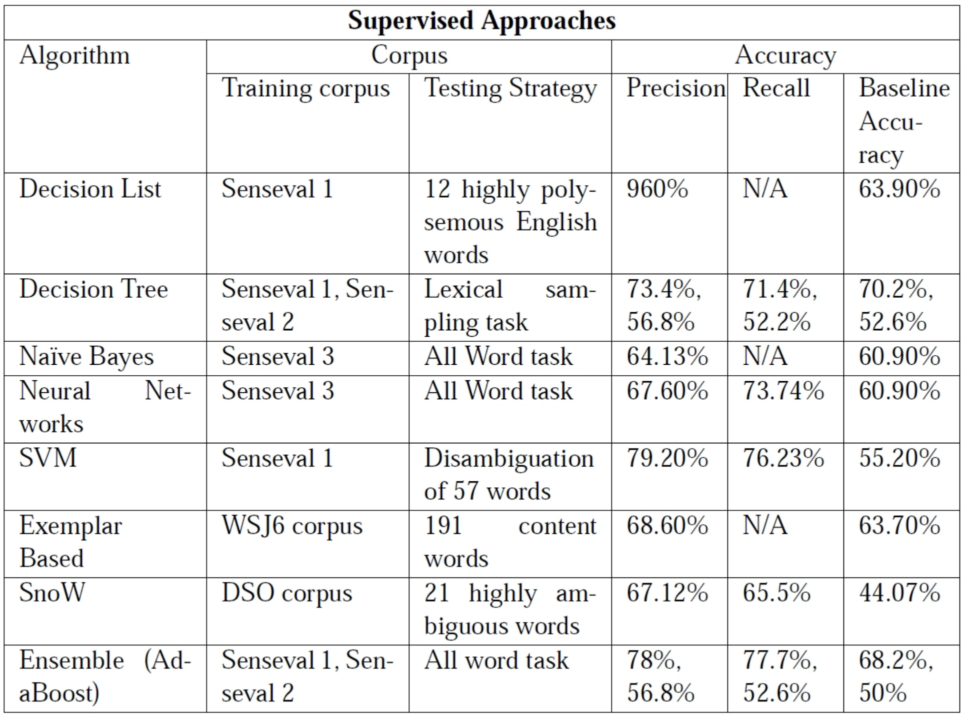
Summary

This section presents the consolidated results of various approaches discussed so far. For simplicity, the results are tabulated showing the accuracy in terms of precision, recall and baseline value wherever possible.

The traininig and testing strategies used along with the corpus and lexical resources used are also shown in the tables against each approach. The accuracy measures are taken as reported in the respective papers, and therefore, some of the figures might be absent based on the figures presented in original papers.

5.1 Supervised Approaches

First, we present the performances of the Supervised Approaches. In case of Supervised approaches, the training corpus, the testing strategy and the corresponding accuracies have been enlisted in the table given below.



Supervised Approaches

Algorithm	Corpus		Accuracy		
	Training corpus	Testing Strategy	Precision	Recall	Baseline Accuracy
Decision List	Senseval 1	12 highly polysemous English words	960%	N/A	63.90%
Decision Tree	Senseval 1, Senseval 2	Lexical sampling task	73.4%, 56.8%	71.4%, 52.2%	70.2%, 52.6%
Naïve Bayes	Senseval 3	All Word task	64.13%	N/A	60.90%
Neural Networks	Senseval 3	All Word task	67.60%	73.74%	60.90%
SVM	Senseval 1	Disambiguation of 57 words	79.20%	76.23%	55.20%
Exemplar Based	WSJ6 corpus	191 content words	68.60%	N/A	63.70%
SnoW	DSO corpus	21 highly ambiguous words	67.12%	65.5%	44.07%
Ensemble (AdaBoost)	Senseval 1, Senseval 2	All word task	78%, 56.8%	77.7%, 52.6%	68.2%, 50%

5.2 Semi-Supervised Approaches

Second, we present the performances of the Semi-Supervised Approaches. In case of Semi-Supervised approaches also, the training corpus, the testing strategy and the corresponding accuracies have been enlisted in the table given below.

5.3 Hybrid Approaches

Last, we present the performances of the Hybrid Approaches. Hybrid approaches also have a training corpus for learning purposes and also testing strategies similar to the above approaches. Its figures are stated in Table given below.

5.4 Eye-tracking

The idea of using eye-tracking for WSD is new and has not been applied in many domains of NLP as well. It has been used in a few instances like examining effects of local lexical predictability on fixation durations and fixation locations during sentence reading *etc.* Our work hence remains novel in its place as such an attempt to link eye-tracking to sense annotation has not been attempted before.

5.5 Wordnet Linking

Our work on linking Hindi Wordnet to its English counterpart, has made use of several approaches used in Korean Wordnet and PanLexicon Project. Some heuristics both in the existing system and new have been inspired by approaches like Maximum Similarity Heuristic, Word match, IS-A relation, Finding the Best Matching Contexts *etc.*

Semi-supervised Approaches					
Algorithm	Corpus		Accuracy		
	Training corpus	Testing Strategy	Precision	Recall	Baseline Accuracy
Decision List (BootStrapping)	Senseval 1	12 highly polysemous English words	96.1%	N/A	63.9%
Monosemous Relatives	World Wide Web	All word task	64.1%	N/A	62.4%

Hybrid Approaches					
Algorithm	Corpus		Accuracy		
	Training corpus	Testing Strategy	Precision	Recall	Baseline Accuracy
IWSD	SemCor	52 texts created from 6 SemCor files	92.2%	55%	N/A
SenseLearner	Senseval 3	All Word task	64.60%	64.60%	60.90%
SSI	Senseval 3	Gloss disambiguation task	68.5%	68.4%	N/A

Bibliography

[Agirre and Martinez, 2000] Agirre, E. and Martinez, D. (2000). Exploring automatic word sense disambiguation with decision lists and the web. Arxiv preprint cs/0010024.

[Boser et al., 1992] Boser, B., Guyon, I., and Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In Proceedings of the fifth annual workshop on Computational learning theory, pages 144–152. ACM.

[Chatterjee et al., 2012] Chatterjee, A., Joshi, S., Bhattacharyya, P., Kanojia, D., and Meena, A. (2012). A study of the sense annotation process: Man v/s machine. International Conference on Global Wordnets.

[Cost and Salzberg, 1993] Cost, S. and Salzberg, S. (1993). A weighted nearest neighbor algorithm for learning with symbolic features. Machine learning, 10(1):57–78.

[Diab and Resnik, 2002] Diab, M. and Resnik, P. (2002). An unsupervised method for word sense tagging using parallel corpora. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, ACL '02, pages 255–262, Morristown, NJ, USA. Association for Computational Linguistics.

[Drieghe et al., 2008] Drieghe, D., Pollatsek, A., Staub, A., and Rayner, K. (2008). The word grouping hypothesis and eye movements during reading.

[Escudero et al., 2000] Escudero, G., Màrquez, L., and Rigau, G. (2000). An empirical study of the domain dependence of supervised word sense disambiguation systems. In Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora, pages 172–180, Morristown, NJ, USA. Association for Computational Linguistics.

[Escudero et al., 2001] Escudero, G., Màrquez, L., and Rigau, G. (2001). Using lazyboosting for word sense disambiguation. In Proceedings of 2nd International Workshop "Evaluating Word Sense Disambiguation Systems", SENSEVAL-2. Toulouse.

[Freund et al., 1999] Freund, Y., Schapire, R., and Abe, N. (1999). A short introduction to boosting. Journal-Japanese Society For Artificial Intelligence, 14(771-780):1612.

[Gonzalo et al., 2003] Gonzalo, J., Verdejo, F., and Chugar, I. (2003). The web as a resource for wsd. In 1st MEANING workshop, Spain. Citeseer.

[Lee et al., 2000] Lee, C., Lee, G., and Yun, S. (2000). Automatic wordnet mapping using word sense disambiguation. In Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics-Volume 13, pages 142–147. Association for Computational Linguistics.

[Li and Li, 2004] Li, H. and Li, C. (2004). Word translation disambiguation using bilingual bootstrapping. Comput. Linguist., 30:1–22.

[McCulloch and Pitts, 1943] McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of mathematical biology, 5(4):115–133.

[Mihalcea and Banea, 2007] Mihalcea, R. and Banea, C. (2007). Learning multilingual subjective language via cross-lingual projections. In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics.

[Mukund et al., 2010] Mukund, S., Ghosh, D., and Srihari, R. K. (2010). Using cross-lingual projections to generate semantic role labeled corpus for urdu: a resource poor language. In Proceedings of the 23rd International Conference on Computational Linguistics, COLING '10, pages 797–805, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Navigli and Velardi, 2005a] Navigli, R. and Velardi, P. (2005a). Structural semantic interconnections: A knowledge-based approach to word sense disambiguation. In IEEE Transactions On Pattern Analysis and Machine Intelligence.

[Navigli and Velardi, 2005b] Navigli, R. and Velardi, P. (2005b). Structural semantic interconnections: a knowledge-based approach to word sense disambiguation. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 27(7):1075–1086.

[Ng et al., 2003] Ng, H. T., Wang, B., and Chan, Y. S. (2003). Exploiting parallel texts for word sense disambiguation: an empirical study. In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1, ACL '03, pages 455–462, Morristown, NJ, USA. Association for Computational Linguistics.

[Quinlan, 1986] Quinlan, J. (1986). Induction of decision trees. Machine learning, 1(1):81–106.

[Quinlan, 1996] Quinlan, J. (1996). Improved use of continuous attributes in c4. 5. Arxiv preprint cs/9603103.

[Rada, 2005] Rada, M. (2005). Large vocabulary unsupervised word sense disambiguation with graph-based algorithms for sequence data labeling. In In Proceedings of the Joint Human Language Technology and Empirical Methods in Natural Language Processing Conference (HLT/EMNLP), pages 411–418.

[Rivest, 1989] Rivest, R. (1989). Inferring decision trees using the minimum description length principle. Inform. Comput, 80:227–248.

[Sammer and Soderland, 2007] Sammer, M. and Soderland, S. (2007). Building a sense-distinguished multilingual lexicon from monolingual corpora and bilingual lexicons. Proceedings of Machine Translation Summit XI, pages 399–406.

[Schapire and Singer, 1999] Schapire, R. and Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. Machine learning, 37(3):297–336.

[Vainio et al., 2009] Vainio, S., Hyona, J., and Pajunen, A. (2009). Lexical predictability exerts robust effects on fixation duration, but not on initial landing position during reading. volume 56, pages 66–74.

[Yarowsky, 1995] Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In Proceedings of the 33rd annual meeting on Association for Computational Linguistics, pages 189–196. Association for Computational Linguistics.

[Yarowsky, 2000] Yarowsky, D. (2000). Hierarchical decision lists for word sense disambiguation. Computers and the Humanities, 34(1):179–186.

[Yarowsky and Ngai, 2001] Yarowsky, D. and Ngai, G. (2001). Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies, NAACL '01, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Zwirello, 2005] Zwirello, C. (2005). Snow applied to word sense disambiguation.