Chapter 1

Word Sense Disambiguation: Literature Survey (June 2012)

In this chapter, we provide a brief overview of the existing work on WSD examined between December 2009 and May 2012. Since our work is focused on the unsupervised techniques, the notable unsupervised approaches are summarized in the end.

1.1 Supervised Algorithms

In the last two decades, 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 notable supervised algorithms for sense disambiguation in the literature.

1.1.1 Decision Lists

The decision lists, first described by Rivest (1987) are a set of rules in an ordered list format. A decision list is a set of weighted *if-then-else* rules. It was first used by Yarowsky (1994) 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 features 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 *One sense per collocation* property, which states that the 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 and is denoted as $Score(S_i)$.

1.1.1.3 The WSD algorithm

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. Formulating the above we have:
$\hat{S} = argmax_{S_i \in Senses_D^{(w)}} Score(S_i)$

Where:
$\hat{S}$ = A candidate sense.
D = A reference Dictionary.

1.1.2 Decision Trees

The decision tree (Quinlan, 1986) 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 tree 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 due to 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*.

1.1.3 Neural Networks

A Neural Network described by (Rumelhart et al., 1994), (Hagan et al., 1996) 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 Model* (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.

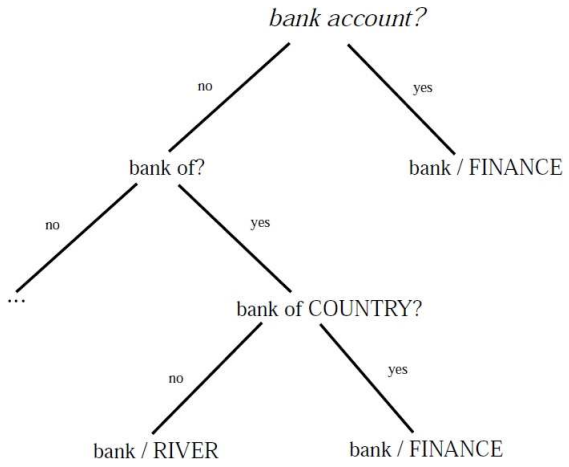

Figure 1.1: An example of Decision Tree

- 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 feed-forward 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 (Ng, 1997) 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 *kNN* 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

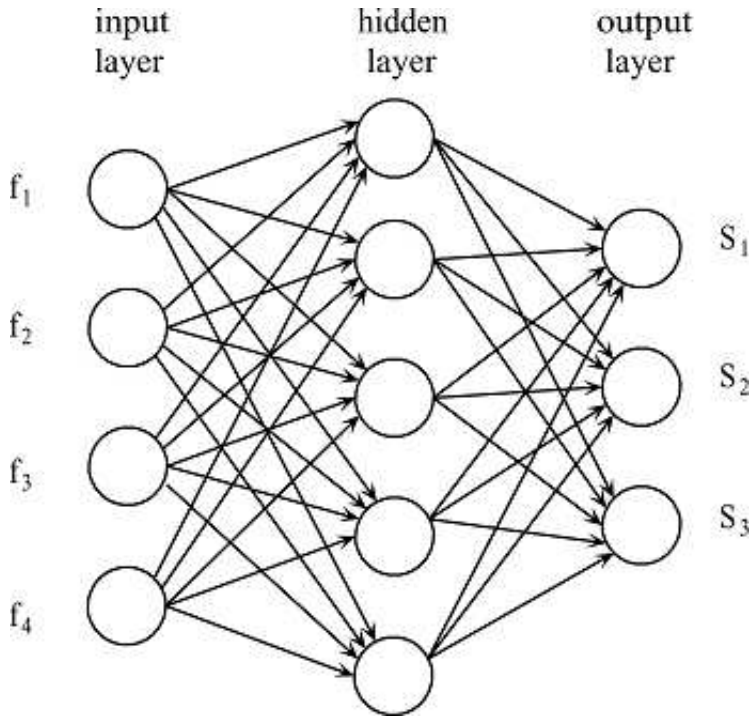

Figure 1.2: An example of feed-forward network for WSD (Figure courtesy Hagan et al. (1996))

mathematically computed by the distance, *e.g.*, the *Hamming distance*:

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

where:
w_j : weight of the j^{th} 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 is determined experimentally. Feature weights w_j can be estimated, *e.g.*, with the *gain ratio measure*. Complex metrics, like the *modifed value difference metric*, 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

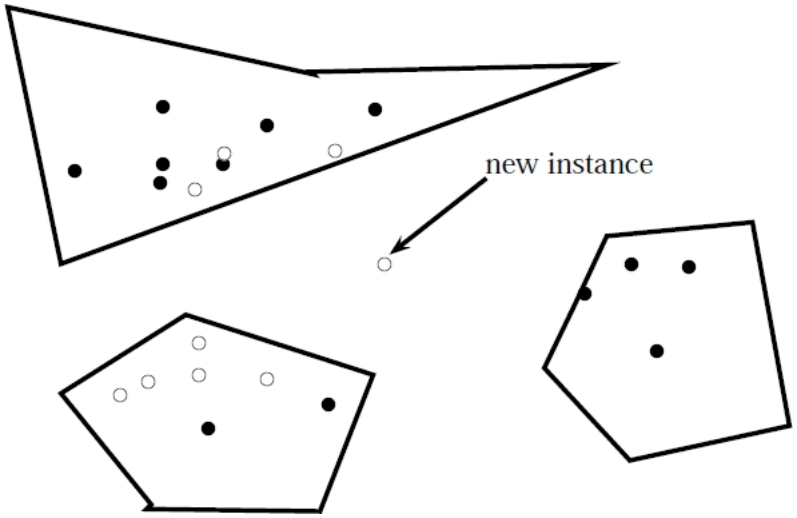

Figure 1.3: An example of kNN on 2D plane (Figure courtesy Ng (1997))

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, 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* (Margineantu and Dietterich, 1997) 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 extension of AdaBoost which deals with multiclass, multilabel classification is AdaBoost.MH as demonstrated by Abney et al. (1999). An application of AdaBoost called *LazyBoosting* was also used by Escudero et al. (2001). LazyBoosting is essentially AdaBoost used for WSD purpose.

1.1.6 Support Vector Machines

Support Vector Machines were introduced by Hearst et al. (1998) 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. Figure 1.1.6 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 here. 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.

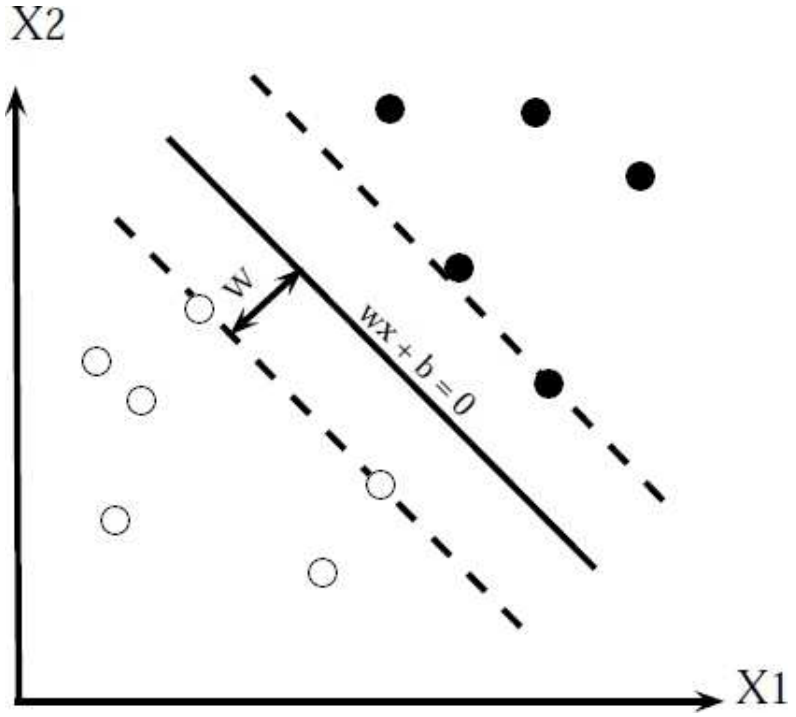

Figure 1.4: The geometric intuition of SVM (Figure courtesy Hearst et al. (1998))

1.1.7 SNoW Architecture

Snow stands for ***Sparse Network Of Winnows***, which is an online learning algorithm. The fundamental construct of the algorithm is the *Winnow algorithm* (Blum, 1995). The algorithm learns very fast in the presence of many binary input features, as it consists of a linear threshold algorithm and updates multiplicative weight for problems having 2 classes (Carlson et al., 1999).

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, SNoW behaves somewhat like a neural net, which takes features as input and outputs the class with the highest activation value. According to Blum (1995), 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 (1992), 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 probability.

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 disambiguating 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 up to 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 collection of words as corpus. This enables the algorithms to have an automatically annotated corpus, which has tremendously huge size, the *web corpus*.
Monosemous relatives approach 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 $\in$ 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 necessary for such approaches.

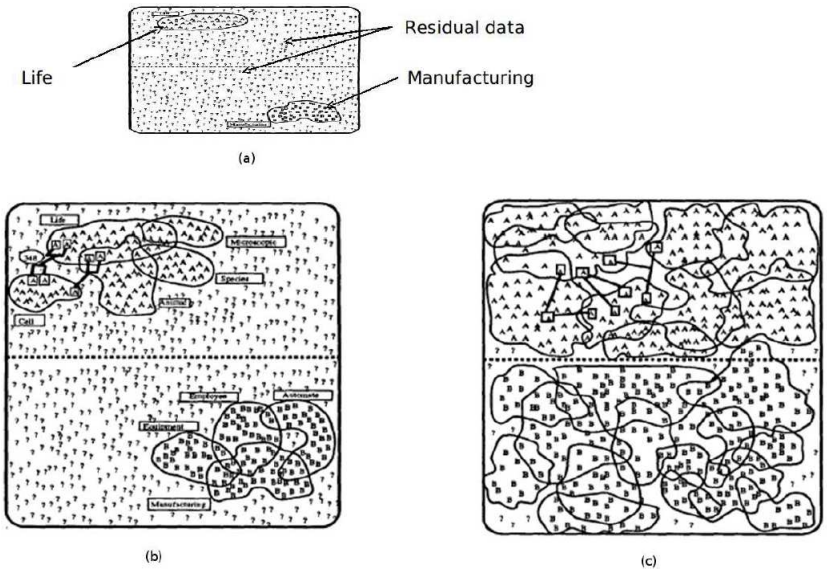

Figure 1.5: figure showing growth of Semi-supervised decision list on two senses of plant *viz*., life and manufacturing. (a) The initial seed data. (b) Growth of the seed set. (c) Seed data converges. (Figure courtesy Yarowsky (1992))

1.3 Unsupervised algorithms

A Supervised approach in WSD needs training data on which it builds models or hypotheses. The training data has to be manually created, which is very expensive, both temporally and financially. This problem is typically known as Knowledge acquisition bottleneck. Unsupervised algorithms overcome this problem by assuming that the sense of a word will depend on those of neighboring words. The *single common thread* which binds these algorithms is the *clustering strategy used on the words* in the un-annotated corpus. The words are then classified into one of these clusters based on some similarity measure. These algorithms are therefore termed as *Word sense discrimination algorithms* rather than disambiguation algorithms. Although they do not end up finding the actual sense of a word, the clustering and classification enables one to label the senses, and therefore these approaches are treated as part of WSD.

Since the sense clusters derived by these algorithms may not match the actual senses defined in Lexical resources like dictionaries, the evaluation of these algorithms needs to be carried out manually, by asking language experts to corroborate the results. Based on the *type of clustering performed* by unsupervised algorithms, they can be classified as follows:

1.3.1 Context clustering algorithms

Context is formally a discourse that surrounds a language unit (*e.g.* a word) and helps to determine its interpretation. The algorithms in this domain represent the occurrences of target words as word vectors. From these vectors, context vectors are formed and meaning similarity is found that is a function of cosine between the context vectors:

$$sim(v,w) = \frac{v \cdot w}{|v| \cdot ||w|} = \frac{\sum_{i=1}^{m} v_i \cdot w_i}{\sqrt{\sum_{i=1}^{m} v_i^2} \sum_{i=1}^{m} w_i^2}$$

Schütze (1992) formulated a way to represent the vector space with words as dimensions. Arbitrary words can be chosen as axes, and the words in the corpus can be vectorized based on the counts of co-occurrences of these words with each of the axes. The occurrence of every word within a window size of *k* is counted.

The following example shows axes as bank and house, with the context words for interest (x_1, y_1), money (x_2, y_2), deposits (x_3, y_3), and door (x_4, y_4). The number in the bracket show the number of times a word occurs with house (x_i) and with bank (y_i) respectively. The words with cosine value of 0 are treated as completely unrelated, whereas, the ones with value 1 are termed synonymous and so on.

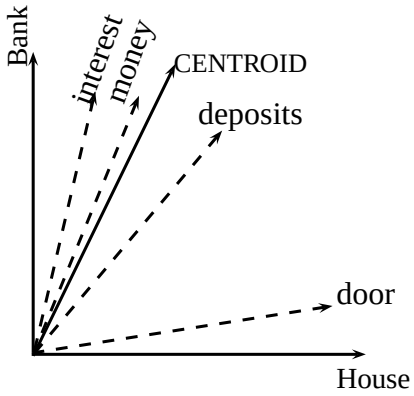

Figure 1.6: An example of word vectors and context vector for stock, calculated as the centroid (or the sum) of the vectors of words occurring in the same context

1.3.1.1 Latent Semantic Analysis

The number of dimensions in the above approach could reach up to a few thousands, and Singular Value Decomposition (SVD) is used to reduce the number of dimensions to around 100. This is done by computing:

$$M = U\Sigma V^*$$

where, M is the m-by-n feature matrix, U is an m-by-m unitary matrix over K, the matrix Σ is m-by-n diagonal matrix with nonnegative real numbers on the diagonal, and V^* denotes the conjugate transpose of V. The diagonal entries in Σ are known as the singular values of M.

Since the original dimensions are largely dependent on each other, and can be approximated as a linear combination of many of the other, the dimensions referring to similar meanings can be merged.

The same procedure as mentioned above is applied to find out words with similar meaning. In order to cluster the context words, a context vector is built as the centroid

of the word vectors which were found in the target context. The centroid finds the approximation of semantic context. It can be seen that the centroid vector is a second order vector, as it does not represent the context directly. In the above figure, the centroid vector is shown for 3 words *viz*., interest, money, and deposits.

1.3.1.2 Context Group Discrimination

This algorithm, which is due to Schütze (1992), goes one step ahead to discriminate the word senses after their context vectors are formed. This algorithm was developed to cluster the senses of the words for which ambiguity is present in the corpus. The algorithm represents senses, words, and context in a multi-dimensional real-valued vector space.
The clustering is done based on contextual similarities between the occurrences. The contextual similarities are still found with cosine function, but the clustering is done using Expectation Maximization algorithm, an iterative, probabilistic model for maximum likelihood estimation.

In the sense acquisition phase, the contexts of all the occurrences of the ambiguous words are represented as context vectors as explained earlier, and a method called average agglomerative clustering is used. The similarity is calculated as a function of number of neighbors common to the words. The more similar words appear in the two contexts, more similar the contexts become. After this, the occurrences are grouped so that occurrences with similar contexts are assigned to same cluster.

A very similar approach is followed in Structural Semantic Interconnections (hybrid algorithm).

1.3.2 Word Clustering Approaches

Context vectors previously explained, are second-order representations of word senses, as in they represent the senses indirectly. The idea here is to cluster the senses based on word vectors, in order to draw out the semantic relationships between the words.

The notable algorithms in this section are:

1.3.2.1 Lin's approach

Lin (1998) clusters two words if they share some syntactic relationship. More the relation, more close the words are situated in the cluster. Given context words w_1, w_2, $\cdots$,w_n and a target word w, the similarity between w and w_i is determined by the information content of their syntactic features.
The previous approach uses context vectors, which conflate senses of words, and thus, similarity of w with each w_i can not be determined with that approach. Therefore, each word is represented in form of a vector. The information contents are then found out using the syntactic features as mentioned previously.

Example:
The facility will employ 500 new employees.
Here, the word *facility* is to be disambiguated (discriminated). From the corpus, the information content of each subject of employ is determined in terms of the *log likelihood*. Since the sense of *installation* for *facility* has highest similarity with the major four subjects of *employ* (*viz*., org, plant, company, industry), it becomes the winner sense.

Senses of Facility	Subjects of Employ		
installation	word	freq	log likelihood
proficiency	*org*	64	51.7
adeptness	*plant*	14	33.0
readiness	*company*	27	29.9
bathroom/toilet	*industry*	9	15.4
In this case Sense 1 of installation	unit	9	10.2
would be the winner sense.	aerospace	2	6.3

Table 1.1: Table showing working of Lin's approach. The winner sense is highlighted.

1.3.2.2 Clustering by Committee

This algorithm, again proposed by Pantel and Lin (2002), can be viewed as an extension over Lin's original approach to WSD discussed previously. This algorithm follows the same steps up to representing the words as a feature vector.
After this, the algorithm recursively decides the clusters, referred to here as *committees*. Given a set of words *W*, the algorithm uses *average link method* to cluster the words. In each step, the words are clustered based on their similarity to the centroids of the committees, and the words which are not similar are gathered. These words, referred to here as *residue words*, are used to discover more committees.

While disambiguating a word *w*, the word is represented using its feature vector and the most similar committee is found for this word.

The algorithm can be summarized as below:

1. `Find K nearest neighbors (kNN) for each element, for some small value of k.`
2. `Form clusters using the kNN obtained from step 1.`
3. `For every new instance e input to the system, assign it to its nearest cluster, as per average link method.`

Typically, the value of *k* is selected to be between 10 and 20. The elements of each cluster are called a *committee*.

1.3.3 Co-occurrence Graphs

Whereas the previous techniques use vectors to represent the words, the algorithms in this domain make use of graphs. Every word in the text becomes a vertex and syntactic relations become edges. The context units (*e.g.* paragraph) in which the target words occur, are used to create the graphs.

The algorithm worth mentioning here is Hyperlex, as proposed by Veronis (2004).

1.3.3.1 Hyperlex

As per this algorithm, the words in context (*e.g.* in the same paragraph) with the target word become vertices, and they are joined with an edge, if they co-occur in same paragraph. The edge weights are inversely proportional to the frequency of co-occurrence of these words.

$$w_{ij} = 1 \text{ - } max\left\{P(w_i|w_j), P(w_j|w_i)\right\}$$

where, $P(w_i \mid w_j) = \frac{Frequency\ of\ co{-}occurrence\ of\ words\ w_i\ and\ w_j}{Frequency\ of\ occurrence\ of\ w_j}$
It can be seen that as an implication, words which co-occur with high frequency, get an edge weight of close to 0 and the other extreme gets 1.

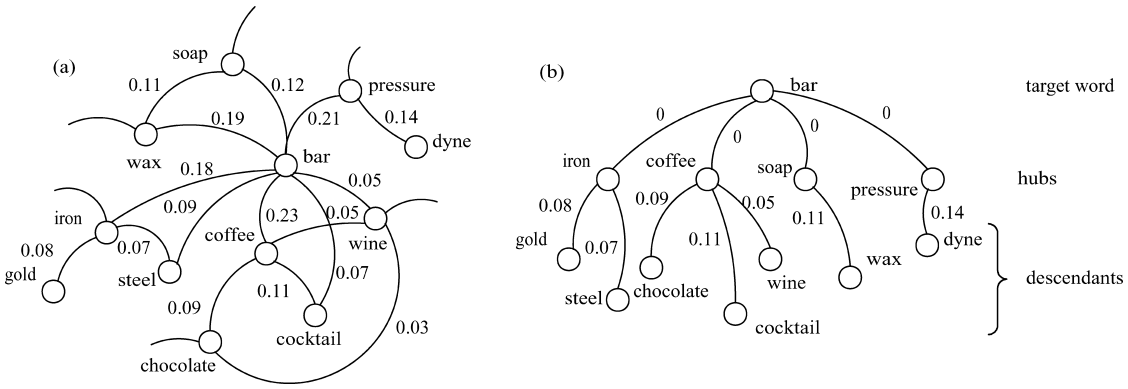

Figure 1.7: Hyperlex showing (a) Part of a co-occurrence graph. (b) The minimum spanning tree for the target word *bar*. (Figure courtesy Navigli (February 2009))

After this is done, iteratively the node with highest relative degree (number of connections) in the graph is selected as a hub. Once this is done, the neighbors of this node cease to be candidates of being hubs. The relative degree for remaining nodes is again computed and this is iterated until the highest relative degree reaches some predefined threshold. The hubs are then linked to the ambiguous word by finding Minimum Spanning Tree (MST) for the resultant graph.

Each node in the MST is assigned a score vector s with as many dimensions as there are components:

$$s = \begin{cases} \frac{1}{1+d(h_{i,v})} & if\ v \in component\ i \\ 0 & otherwise \end{cases}$$

where, $d(h_{i,v})$ is the distance between root hub h_i and node v in the tree. The score vectors of all words are added for the given context. The component with highest score becomes the winner sense.

1.3.4 WSD using parallel corpora

It was experimentally found out that, words in one language, which have multiple meanings, have distinct translations in some other language. This assumption is utilized by Ide et al. (2002) in an algorithm for disambiguation. The algorithm was designed with the aim of obtaining large sense marked corpus automatically annotated with high efficiency.

For this purpose, the algorithm needs raw corpus from more than one language (hence the name parallel corpora). For determining the number of clusters, the algorithm uses a minimum distance computed using:

$$\sqrt{\sum_{i=1}^{n}(v_1(i) - v_2(i))^2}$$

where, v_1 and v_2 are vectors of length n.

The algorithm creates 2 generative models to group and separately model the senses of two languages. The first model, referred to as *Sense model*, groups the words as per senses, irrespective of their language. The second model, referred to as *Concept model*, groups the senses as per their concepts, across both the languages.

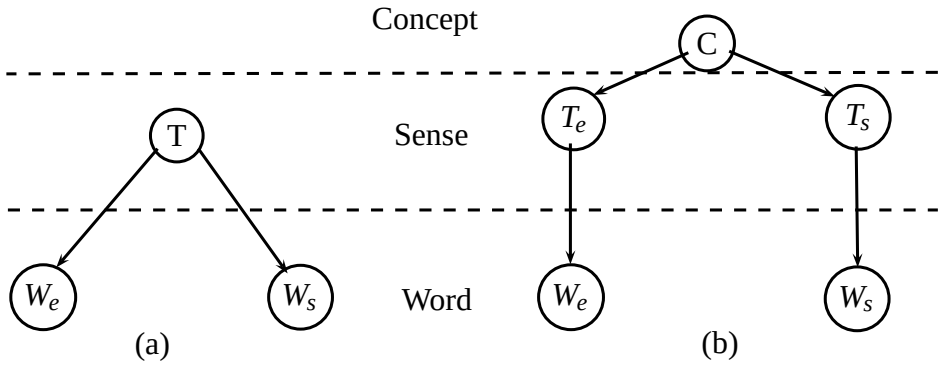

Figure 1.8: Figure showing the (a) Sense model and (b) Concept model (Figure based on works by Ide et al. (2002))

1.3.5 WSD using Roget's Thesaurus categories

Roget's thesaurus is an early Nineteenth century thesaurus which provides classification or categories which are approximations of conceptual classes. This algorithm by Yarowsky (1992) uses precisely this ability of Roget's thesaurus to discriminate between the senses using statistical models. The algorithms observes following:

- Different conceptual classes of words tend to appear in recognizably different contexts.
- Different word senses belong to different conceptual classes.
- A context based discriminator for the conceptual classes can serve as a context based discriminator for the members of those classes.

The algorithm thus identifies salient words in the collective context of the thesaurus category and weighs them appropriately. It then predicts the appropriate category for an ambiguous word using the weights of words in its context. The prediction is done using:

$$\underset{\text{RCat}}{\text{argmax}} \sum_{w \in context} log\,(\frac{Pr(w|RCat)*Pr(RCat)}{Pr(w)})$$

where, *RCat* is the Roger's thesaurus category.

The following table shows the implementation of Yarowsky's algorithm on the target word *crane*. A crane might mean a machine operated for construction purpose (Roget's category of TOOLS/MACHINE) or a bird (Roget's category of ANIMAL/INSECT). By finding the context words for word crane and finding how much weight (similarity) they impose on each sense of crane, the winner sense is selected.

TOOLS/MACHINE	*Weight*	ANIMAL/INSECT	*Weight*
lift	2.44	Water	0.76
grain	1.68		
used	1.32		
heavy	1.28		
Treadmills	1.16		
attached	0.58		
grind	0.29		
Water	0.11		
TOTAL	11.30	*TOTAL*	0.76

Table 1.2: Example list showing the a run of Yarowsky's algorithm for senses of crane belonging to (a) TOOLS/MACHINE and (b) ANIMAL/INSECT domains along with weights of context words. The highlighted sense is the winner sense.

1.4 Eye-tracking

For our experiments pertaining to finding out the role of the context, 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.

1.5 Summary of notable Unsupervised WSD approaches

1.5.1 Monolingual WSD

Depending on the type of evidence or knowledge sources used, existing algorithms for monlingual WSD can be classified into two broad categories, *viz.*, knowledge based approaches and machine learning based approaches. Machine learning based approaches can be further divided into supervised (require sense tagged corpus), unsupervised (require untagged corpus) and semi-supervised approaches (bootstrap using a small amount of tagged corpus and a large amount of untagged corpus).

Knowledge based approaches to WSD such as Lesk (1986), Walker and Amsler (1986), conceptual density by Agirre and Rigau (1996) and random walk algorithm by Rada (2005) essentially do Machine Readable Dictionary lookup. However, these are fundamentally *overlap based* algorithms which suffer from overlap sparsity, dictionary definitions being generally small in length.

Supervised learning algorithms for WSD are mostly word specific classifiers, *e.g.*, Lee Yoong K. and Chia (2004), Ng and Lee (1996) and Yarowsky (1994). The requirement of a large training corpus renders these algorithms unsuitable for resource scarce languages.

Semi-supervised and unsupervised algorithms do not need large amount of annotated corpora, but are again word specific classifiers, *e.g.*, semi-supervised decision list algorithm by Yarowsky (1995) and Hyperlex by Veronis (2004). Hybrid approaches like WSD using Structural Semantic Interconnections as shown in Navigli and Velardi (2005) use combinations of more than one knowledge sources (WordNet as well as a small amount of tagged corpora). This allows them to capture important information encoded in Fellbaum (1998) as well as draw syntactic generalizations from minimally tagged corpora. *These methods which combine evidence from several resources seem to be most suitable in building general purpose broad coverage disambiguation engines and are the motivation for our work.*

1.5.2 Bilingual WSD

The limited performance 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) approaches not using parallel corpora.

The approaches which use parallel corpora rely on the paradigm of *Disambiguation by Translation*, described in the works of Gale et al. (1992), Dagan and Itai (1994), Resnik and Yarowsky (1999), Ide et al. (2001), Diab and Resnik (2002), Ng et al. (2003), Tufis et al. (2004), Apidianaki (2008). 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.

In this work, we are more interested in the second kind of approaches which do not use parallel corpora but rely purely on the in-domain corpora from two (or more) 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 our work which focuses on the broader task of all-words WSD.

Another approach worth mentioning here is the one proposed by Kaji and Morimoto (2002) which aligns statistically significant pairs of related words in language L_1 with their cross-lingual counterparts in language L_2 using a bilingual dictionary. This approach is based on two assumptions (i) words which are most significantly related to a target word provide clues about the sense of the target word and (ii) translations of these related words further reinforce the sense distinctions. The translations of related words thus act as cross-lingual clues for disambiguation. This algorithm when tested on 60 polysemous words (using English as L_1 and Japanese as L_2) delivered high accuracies (coverage=88.5% and precision=77.7%). However, when tested in an all-words scenario, the approach performed way below the random baseline.

Bibliography

[Abney et al.1999] S. Abney, R.E. Schapire, and Y. Singer. 1999. Boosting applied to tagging and pp attachment. In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, volume 130, pages 132–134.

[Agirre and Rigau1996] Eneko Agirre and German Rigau. 1996. Word sense disambiguation using conceptual density. In In Proceedings of the 16th International Conference on Computational Linguistics (COLING).

[Apidianaki2008] Marianna Apidianaki. 2008. Translation-oriented word sense induction based on parallel corpora. In LREC.

[Blum1995] A. Blum. 1995. Empirical support for winnow and weighted-majority algorithms: results on a calendar scheduling domain. In Machine Learning. Citeseer.

[Carlson et al.1999] A. Carlson, C. Cumby, J. Rosen, and D. Roth. 1999. The snow learning architecture. Technical report, Technical report UIUCDCS.

[Chatterjee et al.2012] Arindam Chatterjee, Salil Joshi, Pushpak Bhattacharyya, Diptesh Kanojia, and Akhlesh Meena. 2012. A study of the sense annotation process: Man v/s machine. International Conference on Global Wordnets, January.

[Dagan and Itai1994] Ido Dagan and Alon Itai. 1994. Word sense disambiguation using a second language monolingual corpus. Comput. Linguist., 20:563–596, December.

[Diab and Resnik2002] Mona Diab and Philip Resnik. 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] D Drieghe, A Pollatsek, A Staub, and K Rayner. 2008. The word grouping hypothesis and eye movements during reading.

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

[Fellbaum1998] C Fellbaum. 1998. Wordnet:an electronic lexical database. The MIT Press.

[Gale et al.1992] William Gale, Kenneth Church, and David Yarowsky. 1992. A method for disambiguating word senses in a large corpus. Computers and the Humanities, 26:415–439. 10.1007/BF00136984.

[Hagan et al.1996] M.T. Hagan, H.B. Demuth, M.H. Beale, and Boulder University of Colorado. 1996. Neural network design. PWS Pub.

[Hearst et al.1998] M.A. Hearst, ST Dumais, E. Osman, J. Platt, and B. Scholkopf. 1998. Support vector machines. Intelligent Systems and their Applications, IEEE, 13(4):18–28.

[Ide et al.2001] Nancy Ide, Tomaž Erjavec, and Dan TufiŞ. 2001. Automatic sense tagging using parallel corpora. In In Proceedings of the 6 th Natural Language Processing Pacific Rim Symposium, pages 212–219.

[Ide et al.2002] Nancy Ide, Tomaz Erjavec, and Dan Tufis. 2002. Sense discrimination with parallel corpora. In Proceedings of the ACL-02 workshop on Word sense disambiguation, pages 61–66, Morristown, NJ, USA. Association for Computational Linguistics.

[Kaji and Morimoto2002] Hiroyuki Kaji and Yasutsugu Morimoto. 2002. Unsupervised word sense disambiguation using bilingual comparable corpora. In Proceedings of the 19th international conference on Computational linguistics - Volume 1, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Lee Yoong K. and Chia2004] Hwee T. Ng Lee Yoong K. and Tee K. Chia. 2004. Supervised word sense disambiguation with support vector machines and multiple knowledge sources. In Proceedings of Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, pages 137–140.

[Lesk1986] Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In In Proceedings of the 5th annual international conference on Systems documentation.

[Li and Li2004] Hang Li and Cong Li. 2004. Word translation disambiguation using bilingual bootstrapping. Comput. Linguist., 30:1–22, March.

[Lin1998] Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In Proceedings of the 17th international conference on Computational linguistics, pages 768–774, Morristown, NJ, USA. Association for Computational Linguistics.

[Margineantu and Dietterich1997] D.D. Margineantu and T.G. Dietterich. 1997. Pruning adaptive boosting. In MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-, pages 211–218. MORGAN KAUFMANN PUBLISHERS, INC.

[Navigli and Velardi2005] Roberto Navigli and Paolo Velardi. 2005. Structural semantic interconnections: A knowledge-based approach to word sense disambiguation. In IEEE Transactions On Pattern Analysis and Machine Intelligence.

[NavigliFebruary 2009] Roberto Navigli. February 2009. Word sense disambiguation: A survey. In ACM Computing Surveys, Vol. 41, No. 2, Article 10.

[Ng and Lee1996] Hwee Tou Ng and Hian Beng Lee. 1996. Integrating multiple knowledge sources to disambiguate word sense: an exemplar-based approach. In Proceedings of the 34th annual meeting on Association for Computational Linguistics, pages 40–47, Morristown, NJ, USA. Association for Computational Linguistics.

[Ng et al.2003] Hwee Tou Ng, Bin Wang, and Yee Seng Chan. 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.

[Ng1997] H.T. Ng. 1997. Exemplar-based word sense disambiguation: Some recent improvements. In Proceedings of the Second Conference on Empirical methods in natural Language Processing, pages 208–213.

[Pantel and Lin2002] Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 613–619, New York, NY, USA. ACM.

[Quinlan1986] J. R. Quinlan. 1986. Induction of decision trees. Mach. Learn, pages 81–106.

[Rada2005] Mihalcea Rada. 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.

[Resnik and Yarowsky1999] Philip Resnik and David Yarowsky. 1999. Distinguishing systems and distinguishing senses: new evaluation methods for word sense disambiguation. Nat. Lang. Eng., 5:113–133, June.

[Rivest1987] Ronald L. Rivest. 1987. Learning decision lists. In Machine Learning, pages 229–246.

[Rumelhart et al.1994] D.E. Rumelhart, B. Widrow, and M.A. Lehr. 1994. The basic ideas in neural networks. Communications of the ACM, 37(3):87–92.

[Schütze1992] H. Schütze. 1992. Dimensions of meaning. In Supercomputing '92: Proceedings of the 1992 ACM/IEEE conference on Supercomputing, pages 787–796, Los Alamitos, CA, USA. IEEE Computer Society Press.

[Tufis et al.2004] Dan Tufis, Radu Ion, and Nancy Ide. 2004. Fine-grained word sense disambiguation based on parallel corpora, word alignment, word clustering and aligned wordnets. In Proceedings of the 20th international conference on Computational Linguistics, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

[Veronis2004] Jean Veronis. 2004. Hyperlex: Lexical cartography for information retrieval. In Computer Speech and Language, volume 18, pages 18(3):223–252.

[Walker and Amsler1986] D. Walker and R. Amsler. 1986. The use of machine readable dictionaries in sublanguage analysis. In In Analyzing Language in Restricted Domains, Grish-man and Kittredge (eds), LEA Press, pages 69–83.

[Yarowsky1992] David Yarowsky. 1992. Word-sense disambiguation using statistical models of roget's categories trained on large corpora. In Proceedings of the 14th conference on Computational linguistics, pages 454–460, Morristown, NJ, USA. Association for Computational Linguistics.

[Yarowsky1994] David Yarowsky. 1994. Decision lists for lexical ambiguity resolution: Application to accent restoration in spanish and french. In In Proceedings of the 32nd Annual Meeting of the association for Computational Linguistics (ACL), pages 88–95.

[Yarowsky1995] David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In Proceedings of the 33rd annual meeting on Association for Computational Linguistics, pages 189–196, Morristown, NJ, USA. Association for Computational Linguistics.