A survey on word sense disambiguation

Himanshu Singh[1] and **Pushpak Bhattacharyya**[2]

[1]himanshus096@gmail.com, Computer Science and Engineering, Indian Institute of Technology Bombay
[2]pb@cse.iitb.ac.in, Computer Science and Engineering, Indian Institute of Technology Bombay

June 30, 2019

1 Introduction

Sentence 1 : Getting rid of *cricket* is not a game
Sentence 2 : He played the *cricket* match yesterday

In sentence 1, *cricket* has a sense of *insect* while in sentence 2 *cricket* has a sense of *sport played by eleven players*. Thus for a given word there can be more than one sense. Words with more than one senses are called polysemous words while words with single sense are called monosemous word.
Consider following sentences:

He is standing near the **bank** of a river.
His **bank** account details are compromised.

In the first sentence, sense of bank is *a sloped land near a river* and in the second sentence, sense of bank is *financial institution*. So, depending on the context, sense of the word changes. Let's take another example:

ईश्वर सब की पानी रखते हैं
एक गिलास पानी लाना

ishwar sab ki paani rakhate hain
ek gilas paani lana
In the first sentence, sense of *paani* is *prestige* and in the second sentence, sense of *paani* is *water*. From above example we can conclude that WSD problem is not a problem of any single language. Its present in every language. This task of identifying senses in a sentence is so simple for human that most of the time we don't even realize that we are trying to disambiguate words in a sentence. But computers have to process the raw data and convert it into structured data so that it can derive some semantic meaning. So, as it is said that what is easy for humans is difficult for computers and vice versa, this task of word sense disambiguation is difficult for computers.

Word sense disambiguation task is defined as determining the sense of words in computational manner. This is an AI-complete task, that is, a task whose solution is at least as hard as the most difficult problems in artificial intelligence.

1.1 Importance of WSD in NLP

WSD is one of the most fundamental tasks in NLP. Many applications in NLP directly or indirectly rely on WSD. Sentiment Analysis, Machine Translation, Information Retrieval, Text summarization, Text Entailment, Semantic Role Labeling are some of the main applications in NLP which depend on WSD.

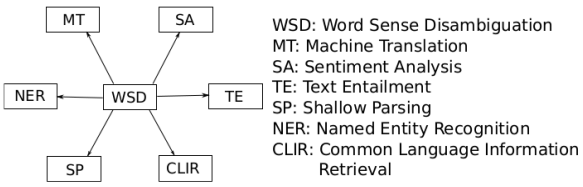

Figure 1: WSD as a heart of NLP

1.2 Formal definition of WSD

Given a piece of text T which contains sequence of words $w_1, w_2..w_n$. Disambiguating each word in the text is all word WSD. Disambiguating a given word is target WSD.

1.3 Why is WSD an AI-complete task?

There are many reasons for WSD to be an AI-complete task:
1.) WSD heavily depends on knowledge soruces which can be labelled or unlabelled data or some semantic networks or some machine readable dictionaries. With time new words appear in the dictionary like google has become synonymous to searching something on the internet. With the addition of new words, solving WSD is even more complex.
2.) fine grained senses vs coarse senses
3.) if text belongs to some domain or it is free text
4.) if set of words to be disambiguated is single or multiple

1.4 Fundamental Problem of WSD task

Knowledge Acquisition Bottleneck : Its very time and money consuming to create a knowledge resource. And every time disambiguation setup i.e. domain, language etc changes the knowledge resource needs to be created again. This is one of the fundamental problem of WSD task.

1.5 Main Elements of WSD Task

There are four main elements of Word Sense Disambiguation task:

1. Selection of Word Senses : A sense inventory partitions the range of meaning of a word into its senses. Word senses cannot be easily discretized, that is, reduced to a finite discrete set of entries, each encoding a distinct meaning. The main reason for this difficulty stems from the fact that the language is inherently subject to change and interpretation.A sense inventory partitions the range of meaning of a word into its senses. Word senses cannot be easily discretized, that is, reduced to a finite discrete set of entries, each encoding a distinct meaning. The main reason for this difficulty stems from the fact that the language is inherently subject to change and interpretation.

2. External Knowledge Sources : Knowledge is a fundamental component of WSD. Knowledge sources provide data which are essential to associate senses with words. They can vary from corpora of texts, either unlabeled or annotated with word senses, to machine-readable dictionaries, thesauri, glossaries, ontologies, etc.

3. Representation of Context : To convert unstructured data into structured data so that computer can analyze it, a preprocessing of the input text is usually performed which involves steps discussed in the Figure 2
If we use neural network to perform WSD task then we may or may not need all these steps. If we have huge amount of data then these steps are not necessary for neural network. We just need to feed word embedding of the words of the sentence and the task of representation of context will be taken care of by the network itself.

4. Choice of a Classification Method : We can choose to use any of the classification methods like supervised, semi-supervised, unsupervised and knowledge based. But if we choose supervised then we need to do corpora labelling with senses. If we choose unsupervised then we will have lots of unlabelled data to train on but its accuracy is not good. If we choose to go with knowledge based then we again need lots of knowledge resources like

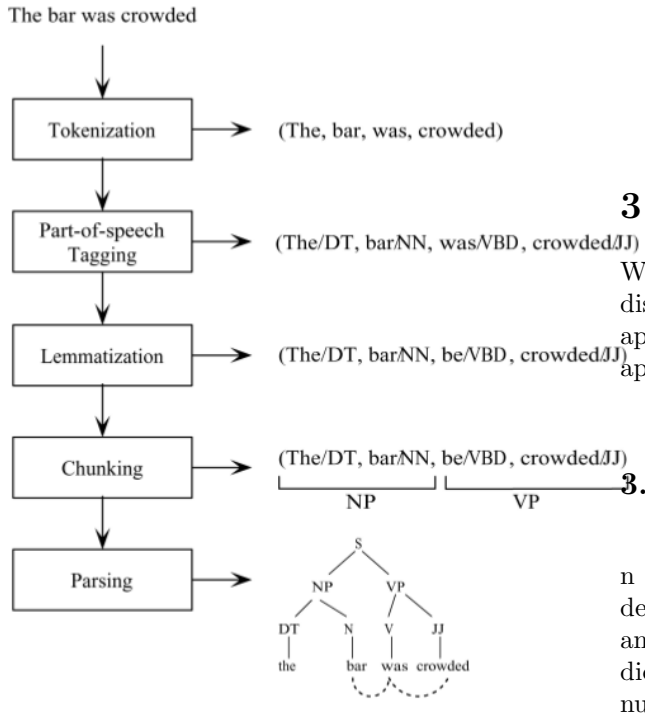

Figure 2: An example of preprocessing steps of text.

ontologies, thesauries etc.

2 Approaches for Word Sense Disambiguation

Approaches for solving word sense disambiguation can be categorized as:

- Supervised approaches : It requires sense tagged corpora.
- Unsupervised approaches : It does not require any tagged corpora. Free text is used to determine the sense of the target word.
- Semi-supervised approaches : It uses limited amount of sense tagged corpora and at the same time it uses lots of unlabelled corpora.
- Knowledge based approaches : It uses huge lexical resources like machine readable dictionaries, ontologies, thesauries etc to determine the sense of the target word.

3 Literature Survey

We have done a thorough study of word sense disambiguation task that involves knowledge based approaches, supervised approaches, semi-supervised approaches and unsupervised approaches.

3.1 Lesk's algorithm : Knowledge based approach

n this approach, the sense of the target word is determinedbased on the overlap among its context and the sense definitions from the machine readable dictionaries. The sense whose gloss has maximum number ofwords in common with the context is assigned to the target word. Each senseof the target word w gets the score as follows:

$$score(S) = |context(w) \cap gloss(S)|$$

This approach is very sensitive to the exact wording in the sense definitions and hence performed poorly.

3.2 Unsupervised approaches

We have done study of unsupervised approaches for WSD that uses Expectation Maximization algorithm and Markov Random Field and Dependency Parser.

1. WSD using Expectation maximization algorithm

Resource Requirement for this algorithm

1. In-domain corpora for two different languages.
2. A synset aligned multilingual dictionary.

Synset Aligned Multilingual Dictionary : This is a type of dictionary where synsets are linked and after that the words within synsets are linked.

Example of Synset Aligned Multilingual Dictionary: Figure 3

Concepts	L1 (English)	L2 (Hindi)	L3 (Marathi)
04321: a youthful male person	{malechild, boy}	{लड़का *(ladkaa)*, बालक *(baalak)*, बच्चा *(bachchaa)*}	{मुलगा *(mulgaa)*, पोरगा *(porgaa)*, पोर *(por)*}

Figure 3: An example of a Multilingual Dictionary

Note that each word in Marathi synset has a corresponding translation word in Hindi synset as shown in Figure 3.

Intuition behind the EM Algorithm
Let there be two languages $L_1 and L_2$. For a given word w in L_2, if a particular sense (say S_1) is more prevalent in a domain (e.g. Health) then a target language (L_1) corpus from the same domain will have more words which are translations of sense S_1 as compared to words which are translations of other senses. Once the sense distributions have been estimated using the EM algorithm, each word in the test corpus is disambiguated by assigning it the most frequent sense as learned from the sense distributions.

2. WSD using MRF and Dependency Parser
This method requires Wordnet, a dependency parser and Stanford POS tagger as knowledge resources. This algorithm is based on two basic ideas:
Sense dependency : Sense of a word depends on sense of other words in the sentence, not the words themselves.
Selective dependency : Sense of a word depends on sense of only few other words in the sentence, not all.
Sense dependency example:
Sentence : *He is standing near the river bank.*
Here word *bank* has got the sense of *sloped land* and not the sense of *financial institution to withdraw money.* It is because of the sense of the context word *river.* Selective Dependency example :
Sentence : *He banks on me that's why he gave me his ATM pin and ATM card to withdraw money.* Sense of the word *bank* which is *trust* depends on the deep semantic of the sentence and not on all the words. Other words like *withdraw* or *money* will cause the word *bank* to disambiguate as *financial institution.* Hence sense of a word does not depend on all the words of the sentence and sometimes it does not depend on any word but the deep semantic of the sentence.

Authors are finding this dependency using dependency parser

Algorithm
Here we are trying to solve all-word WSD. It is done by maximizing the joint probability of all the words senses in the sentence. Finding this joint probability can be intractable. So dependency parser is used to simplify the graph.
Algorithm can be divided into two parts :
1.) Construction of Markov Random Field (MRF)
2.) Maximizing the joint probability using MAP Inference Query
Construction of Markov Random Field: To construct the MRF we need to find nodes and nodes potentials. Along with that we also need to find out edges between nodes and corresponding edge potentials.

Construction of Markov Random Field
How to find nodes and nodes potentials? :
Step 1 : Sentence is fed to Stanford Parts of Speech Tagger to determine the parts of speech tag for each word in the sentence.
Step 2 : Nouns, verbs, adjectives and adverbs are content words. A node is created for each content word in the input sentence. Words have many possible meaning. Similarly node will take different values as senses.
Step 3 : Probability distribution of senses of content word depicts the node potential of the corresponding node in the Markov Random Field. Node potential is determined by the frequency of each sense of each content word. To calculate frequency, Wordnet is used.

Determining Edges and Edge Potentials :

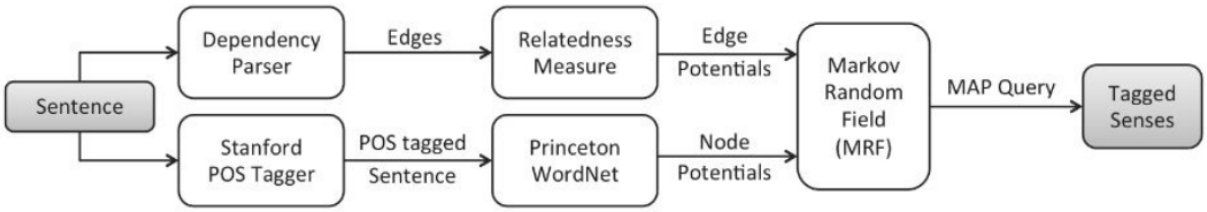

Figure 4: Block diagram of MRF and Link Parser algorithm.

Step 1 : Now the sentence is fed to the Stanford Dependency parser to find out which words are linked to other for its sense disambiguation.
Step 2 : Whichever words are linked in the output of the Stanford Dependency Parser have edges in the MRF.
Step 3 : Two words which are linked, their probability of sense co-occurrence is edge potential of MRF.

Maximizing the joint probability using MAP Inference Query
Let content words of the input sentence be $W = w_1, w_2, ...w_n$ and their senses be $X = x_1, x_2, ...x_n$, respectively. Sense of each word x_i can take k_i possible values from the set $Y_i = y_{i_1}, y_{i_2}...y_{k_i}$, which are all the senses of the word w_i given its POS tag.

Senses are obtained from WordNet.

Example : "Bank is a type of financial organization."
Content words are 'bank', 'is', 'type' and 'financial', 'organization'.
So, w_1 = Bank, w_2 = is, w_3 = type, w_4 = financial, w_5=organization.
x_1 can take two possible values given that it is a noun
$Y_1 = y_1^1(sloped_land), y_1^2(financial_institution)$

$\Psi(x_i)$: Frequency of occurrence of each sense of a word is used to calculate the node potential.

$$\Psi(x_i = y_i^a) \propto frequency(y_i^a) + 1 \forall a$$

$\Psi((x_i), (x_j))$: Edge Potential of edge between x_i and x_j . Co-occurrence of senses of words x_i and x_j is used to determine the edge potential.

$$\Psi(x_i = y_i^a, x_j = y_j^b) \propto M(y_i^a, y_j^b) \forall a, b$$

Normalization of node and edge potentials are done to make it a probability distribution. Joint probability of senses of words in the sentence is given by:

$$\Psi(X) = \Psi(x_1, x_2, ...x_n)$$

$$= \prod_{x_i \in X} \Psi(x_i) \prod_{(x_i, x_j) \in E} \Psi(x_i, x_j)$$

And the problem is reduced to :

$$_Y \Psi(X = Y)$$

Reasoning behind the proposed algorithm
Senses of dependent words affect each other. There is no causal -effect relationship. So, Undirected Graphical Model is used.

Majority of sense dependency is captured in the syntactic structure of the sentence. Dependency Parser does this task.
E.g. "There is a bank which is a financial institution near the river."
Link parser is used to prevent the sense drift that could have been caused because of the presence of the word 'river'
NOTE : Exact inference algorithm (complexity is exponential to the size of largest clique in triangulated graph) can be used as size of the graph formed is very small.

3.3 Supervised approaches

This approach needs sense labelled corpora. Generally used sense tagged corpora for training are SemCor and OMSTI(One Million Sense Tagged Instances). We will discuss here two neural network model for the supervised approach.

1. WSD using synset embeddings

Basic principle : 1.) words are sums of their lexemes and 2.) synsets are sums of their lexemes.

e.g. the embedding of the word bloom is a sum of the embeddings of its two lexemes bloom(organ) and bloom(period); and the embedding of the synset flower-bloom-blossom(organ) is a sum of the embeddings of its three lexemes flower(organ), bloom(organ) and blossom(organ).

Example

Sentence : **He is standing near the bank of the river.**

Word embeddings of words in the sentence:
E_{stand} = [s1, s2, s3, s4, s5, s6, s7, s8, s9, s10]
E_{near} = [n1, n2, n3, n4, n5, n6, n7, n8, n9, n10]
E_{bank}= [b1, b2, b3, b4, b5, b6, b7, b8, b9, b10]
E_{river} = [r1, r2, r3, r4, r5, r6, r7, r8, r9, r10]

Word to be disambiguated : **bank**

Centroid of the sentence $c = (si + ni + bi + ri)$ for $1 \leq i \leq 10$

Senses of bank = "financial institution", "sloping land"

Synset embeddings of "financial institution" $s^{(1)}$ = [f1, f2, f3, f4, f5, f6, f7, f8, f9, f10]

Synset embeddings of "sloping land" $s^{(2)}$ = [l1, l2, l3, l4, l5, l6, l7, l8, l9, l10]

S-cosine feature : ¡cos(c,$s^{(1)}$), cos(c,$s^{(2)}$)¿

S-product feature : ¡$c_1 s_1^{(1)}..c_{10}s_{10}^{(1)}$, $c_1 s_1^{(2)}..c_{10}s_{10}^{(2)}$¿

S-raw feature : ¡$c_1..c_10$, $s_1^{(1)}..s_{10}^{(1)}$, $s_1^{(2)}..s_{(10)}^{(2)}$¿

To test the performance, authors run these features on IMS (A supervised based WSD system). IMS implements three standard WSD feature sets: part of speech (POS), surrounding word and local collocation. They added S-cosine, S-product and S-raw feature along with standard features in IMS to get 1 percent gain in WSD task accuracy.

2. One Single Deep Bidirectional LSTM Network for Word Sense Disambiguation of Text Data

It is a supervised WSD model that leverages a Bidirectional Long Short-Term Memory (BLSTM) network. This network works with sense embeddings, which are learned during model training, and employs word embeddings, which are learned through an unsupervised deep learning approach called GloVe (Global Vectors for word representation) for the context words. This paper uses the idea of sense-context cosine similarities in the model.

Model Description

Input to the network

Left context of a word is fed to the left side par of the BLSTM. One-hot encoding of the context of the target word is fed to the network which undergoes dot product with the pre trained GloVe vectors to give the actual word embedding of the target word context. Cosine similarity of these word embeddings is measured with sense embedding of the target word. Initially, sense embedding of target word is initialized randomly and later with backpropagation of error, network learns the sense embedding of the target word.

Output of the network

When we train the network, for an instance with the correct sense and the given context as inputs, $\hat{y}_s$ is set to be 1.0, and for incorrect senses they are set to be 0.0. During testing, however, among all the senses, the output of the network for a sense that gives the highest value of $\hat{y}_s$ will be considered as the true sense of the ambiguous term

Hidden Layer of the network

The hidden layer h_{cl} is computed as:

$$h_{cl} = ReLU(W_h . \left[h^L_{C_{-1}}; h^R_{C_{+1}}\right] + b_h)$$

$\left[h^L_{C_{-1}}; h^R_{C_{+1}}\right]$ is the concatenated outputs of the right and left traversing LSTMs of the BLSTM when the last context components are met. W_h and b_h are the weights and bias for the hidden layer.

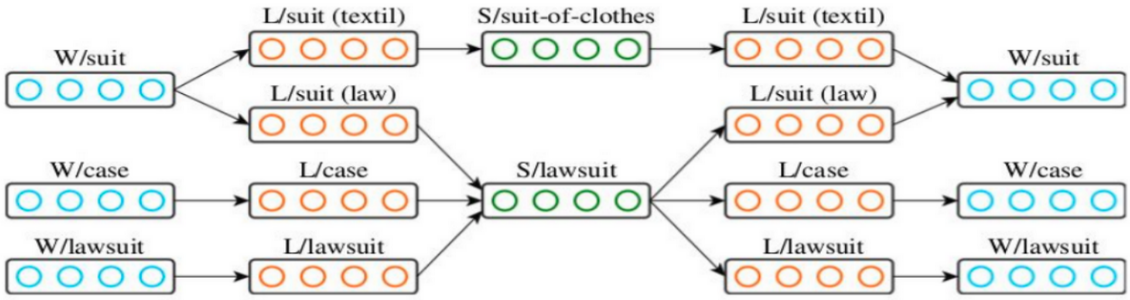

Figure 5: Autoencoder model for generating sysnset embedding as a sum of lexeme embeddings

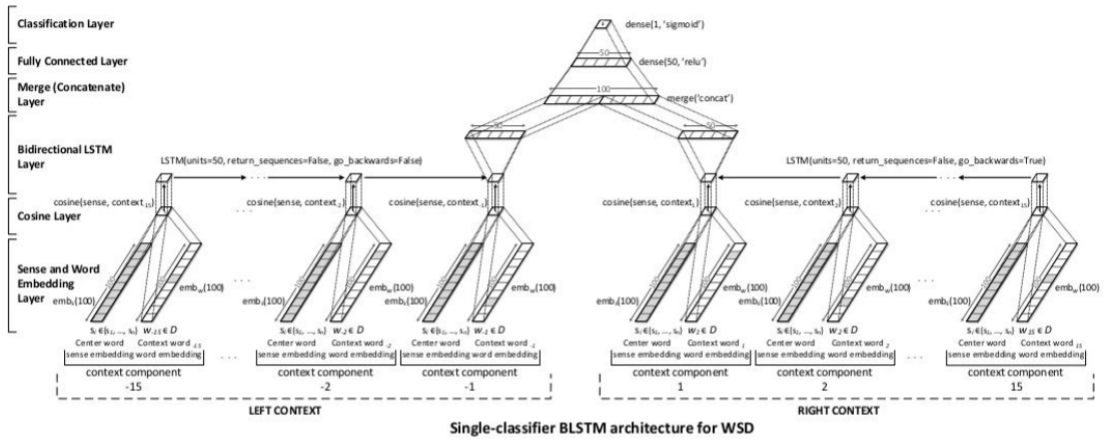

Figure 6: BLSTM network for the proposed model.

3.4 Semi-supervised approach

LSTM Language Model is trained which predicts a held-out word in a sentence.

Algorithm

Step1 : Replace the held-out word with a special symbol $.

Step2 : After LSTM consumes the other words in the sentence, when EOS token is being consumed, project the h (=2048) dimensional hidden layer to p (=512) dimensional context layer. (Word embedding = 512 dimension)

Step3 : Predict the held-out word by applying softmax.

Similarity between two context is computed by the overlap between their bags of predicted words. The top predictions for the query overlap most with the LSTM predictions for '*sense*#1′ —we predict that '*sense*#1′ is the correct sense.

This bag of predictions, while easily interpretable, is just a discrete approximation to the internal state of the LSTM when predicting the held out word. Therefore, the LSTM's context layer from which the bag of predictions was computed is directly used as a representation of the context. Given context vectors extracted from the LSTM, the supervised WSD algorithms classify a word in a context by finding the sense vector which has maximum cosine similarity to the context vector. Sense vector is calculated by averaging context vectors of all training sentences of

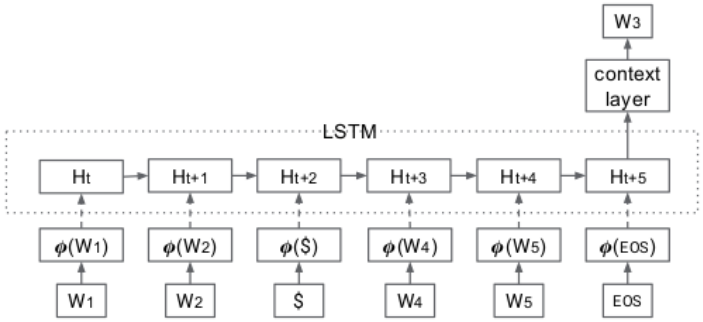

Figure 7: LSTM: Replace the focus word w_3 with a special symbol $ and predict w_3 at the end of the sentence.

id	sentence	top 10 predictions from LSTM	sense
1	Employee compensation is offered in the form of cash and/or *stock*.	cash, stock, equity, shares, loans, bonus, benefits, awards, equivalents, deposits	sense#1
2	The *stock* would be redeemed in five years, subject to terms of the company's debt.	bonds, debt, notes, shares, stock, balance, securities, rest, Notes, debentures	
3	These stores sell excess *stock* or factory overruns .	inventory, goods, parts, sales, inventories, capacity, products, oil, items, fuel	sense#2
4	Our soups are cooked with vegan *stock* and seasonal vegetables.	foods, food, vegetables, meats, recipes, cheese, meat, chicken, pasta, milk	sense#3
query	In addition, they will receive *stock* in the reorganized company, which will be named Ranger Industries Inc.	shares, positions, equity, jobs, awards, representation, stock, investments, roles, funds	?

Figure 8: Top predictions of 'stock' in 5 sentences of different word senses

the same sense.

Semi supervised approach

To overcome drawbacks of supervised algorithm, paper presents a semi-supervised method which augments the labeled example sentences with a large number of unlabeled sentences from the web. Sense labels are then propagated from the labeled to the unlabeled sentences. Adding a large number of unlabeled sentences allows the decision boundary between different senses to be better approximated.

Label propagation (LP) iteratively computes a distribution of labels on the graph's vertices to minimize a weighted combination of:

1. The discrepancy between seed labels and their computed labels distributions.

2. The disagreement between the label distributions of connected vertices.

3. A regularization term which penalizes distributions which differ from the prior (by default, a uniform distribution).

A graph is constructed for each lemma:

1. Labelled vertices are obtained from the labelled sentences which has this lemma.

2. Unlabelled vertices are obtained from the unlabelled sentences from the additional corpus which has this lemma.

Edge : Vertices for sufficiently similar sentences are connected by an edge whose weight is the cosine similarity between the respective context vectors, using the LSTM language model.

WSD Task : To classify an occurrence of the lemma, an additional vertex is created for the new sentence and run Label Propagation to propagate the sense labels from the seed vertices to the unlabeled vertices.

4 Summary

We studied the word sense disambiguation task in depth which includes understanding the problem associated with the task. We also did a thorough literature survey of supervised, semi-supervised, unsupervised and knowledge based algorithm for solving word sense disambiguation task.

References

[1] Devendra Singh Chaplot, Pushpak Bhattacharyya, and Ashwin Paranjape. Unsupervised word sense disambiguation using markov random field and dependency parser. In *AAAI*, pages 2217–2223, 2015.

[2] Chuong B Do and Serafim Batzoglou. What is the expectation maximization algorithm? *Nature biotechnology*, 26(8):897, 2008.

[3] Mitesh M Khapra, Salil Joshi, and Pushpak Bhattacharyya. It takes two to tango: A bilingual unsupervised approach for estimating sense distributions using expectation maximization. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 695–704, 2011.

[4] Douwe Kiela, Felix Hill, and Stephen Clark. Specializing word embeddings for similarity or relatedness. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2044–2048, 2015.

[5] Rada Mihalcea, Timothy Chklovski, and Adam Kilgarriff. The senseval-3 english lexical sample task. In *Proceedings of SENSEVAL-3, the third international workshop on the evaluation of systems for the semantic analysis of text*, 2004.

[6] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[7] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[8] Roberto Navigli. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10, 2009.

[9] Masataka Ono, Makoto Miwa, and Yutaka Sasaki. Word embedding-based antonym detection using thesauri and distributional information. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 984–989, 2015.

[10] Ahmad Pesaranghader, Ali Pesaranghader, Stan Matwin, and Marina Sokolova. One single deep bidirectional lstm network for word sense disambiguation of text data. *arXiv preprint arXiv:1802.09059*, 2018.

[11] Sascha Rothe and Hinrich Schütze. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. *arXiv preprint arXiv:1507.01127*, 2015.

[12] Kaveh Taghipour and Hwee Tou Ng. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 314–323, 2015.

[13] Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. Semi-supervised word sense disambiguation with neural models. *arXiv preprint arXiv:1603.07012*, 2016.

[14] Zhi Zhong and Hwee Tou Ng. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 system demonstrations*, pages 78–83. Association for Computational Linguistics, 2010.