Integrating Word Embedding based Cluster Features in Hindi Dependency Parsing

by

Silpa Kanneganti, vandan.mujadia , Dipti Misra Sharma

in

18th International Conference on Computational Linguistics and Intelligent Text Processing (*CICLing-2017*)

Budapest, Hungary

Report No: IIIT/TR/2017/-1

Centre for Language Technologies Research Centre
International Institute of Information Technology
Hyderabad - 500 032, INDIA
April 2017

Integrating Word Embedding based Cluster Features in Hindi Dependency Parsing

Silpa Kanneganti, Vandan Mujadia, Dipti M. Sharma

LTRC, International Institute of Information Technology,
Hyderabad

Abstract In this paper we present our efforts at incorporating word embedding cluster based features to aid data driven dependency parsing. This work is motivated by the observation that two words connected to a common head word through an identical dependency relation tend to be close to each other in the semantic space. We present a simple and effective semi-supervised method to introduce features that incorporate dependency label clusters derived from a large annotated corpus into data driven dependency parsing. We demonstrate the effectiveness of the approach in a series of experiments on the Hindi Dependency Treebank.

1 Introduction

Over the past few years, several efforts have been made towards developing robust data driven dependency parsing techniques as well as resources encoding syntactic and semantic features. Syntactic resources although help capture information like speech category of word forms, their morphological features and syntactic relations etc, there is still a gap in deducing and disambiguating some of the features accurately. Hence, the need for richer information invoked several efforts at creating higher order linguistic resources. Previous work on semantic annotation shows that semantic information delivers significant improvements in dependency parsing. Hence more efforts are being made at introducing semantic information into syntactic disambiguation in parsing [1] [2]. Word representations derived from unlabeled text have proven useful for many NLP tasks [3], [4]. Word representations can be discrete [5] or continuous [6]. For the purpose of this work, we experiment with features derived from discrete representations. We conduct all our experiments on Hindi, a MoR-FWO (morphologically rich, relatively freer word order) language although the approach is language independent.

We propose a semi-supervised approach for introducing features derived from dependency label based word embedded clusters into dependency parsing. Given a dependency annotated corpus, we group edges from the parsed trees based on their dependency relation labels. All the edges with common heads and dependency relations from the corpus are grouped together into clusters i.e., for a given token word (W) in the data, all its children in the entire corpus are pooled together based on their dependency label with W. W in this context can be a uni-gram, bi-gram or a tri-gram (n-gram form henceforth) of a chunk head i.e., we extract n-grams of all the chunk heads from the corpus and apply the aforementioned clustering approach to it. These clusters are now projected into a semantic space which inturn consists of n-grams created from a large corpus of raw data in-order to get word vector representations of the corresponding work tokens. The resulting clustered word embeddings are used to make better predictions (transitions or arcs) in data driven dependency parsing. This work is motivated by the distributional semantics hypothesis that, words that share context, have similar meanings. For a free word order language like Hindi, post positions, subject markers, conditionals, emphatic markers, vibhaktis [7] help define relationships between nodes in a dependency tree. Therefore by introducing an n-gram based approach, we aim to capture more context related information.

Through our experiments we show that the features we are proposing help improve the performance of both transition and graph based models, currently two of the prominent approaches in data driven dependency parsing. We achieve an improvement of 0.88 % and 1.03% Labeled attachment score(LAS) for the transition-based and graph-based models respectively, over 2242 train and 891 test sentences from Hindi Dependency Treebank Data (HDTB) [8].

2 Related Work

Our approach is inspired by [9], who demonstrated the effectiveness of using word clusters as features in a discriminative learning approach although they worked on named-entity recognition. Discrete representations are being used for POS tagging [10] [11], named entity recognition [12], supersense tagging [13], grammar induction [14], constituency parsing [15] and dependency parsing [16]. [4] demonstrate improved in-domain dependency parsing using features based on discrete Brown clusters. For dependency parsing, [17] also used embedding features, but they stick to one set of pre-trained embeddings while we train our own, tailored to the task. Attempts have been made to utilize hand annotated semantic information for dependency parsing [18], [19], [20]. [21] utilizes English WordNet semantic classes to help improve parsing accuracies. [22] makes use of Hindi WordNet based features to improve dependency parsing for Hindi Treebank data. [23] proposed methods using word embedding derived features for Dependency Parsing in Hindi.

3 Background and Challenges

Word2Vec [24],[25] and GloVe [26] are popular unsupervised systems used to determine the semantic distances between words. Word vectors are represented as a very high dimensional but sparse vectors in which each entry is a measure of the association between the word and a particular context [27] [28]. In some works, the dimensionality of the sparse word-context vectors is reduced, using techniques such as SVD [29] or

LDA to represent words as dense vectors that are derived by various training methods inspired from neural-network language modeling [6] [24]. These representations, referred to as "neural embeddings" or "word embeddings", have been khown to perform well across a variety of tasks [30] [31]. In this work we have trained word embeddings using continuous bag-of-words (CBOW) and skip-gram (SKIP) models described in [24], implemented in open-source toolkit word2vec.

WordNets are sizable lexical databases in which, nouns, verbs, adjectives and adverbs are categorized into sets of cognitive synonyms (a.k.a. synsets). Each of these sets express a distinct concept. As mentioned in [32], these synsets are interlinked by means of conceptual-semantic and lexical relations. For the purpose of this work we use wordnet [33] for Indian languages.

Transition-based and Graph-based parsing models are two of the most dominant approaches in dependency parsing. Transition-based parsers, conditioned on the parse history, learn a model for scoring transitions from one parser state to the next and perform parsing by greedily taking the highest-scoring transition out of every parser state until they reach a complete dependency graph. Graph-based parsers learn a model for scoring possible dependency graphs for a given sentence, by factoring the graphs into their component arcs, and perform parsing by searching for the highest-scoring graph. While their approaches are quite different, both models have been known to achieve state-of-the-art accuracy on par with each other for a wide range of languages. In this work, we experiment with both approaches and show significant improvements in accuracies in both the cases.

For the purpose of this work, we use dependency corpus data Hindi Treebank (HTB ver-0.51) proposed in [34] [35]. With morphological, part-of-speech and dependency annotations based on the Computational Paninian Framework (henceforth CPG)[8], [7]. In the dependency annotation, relations are mainly verb-centric. The relationship between a verb and its arguments is called a 'karaka'. Besides karaka relations, dependency relations also exist between two nouns (genitives), nouns and their modifiers (adjectival modification, relativization) and between verbs and their modifiers (adverbial modification including subordination). CPG essentially provides a syntacticosemantic dependency annotation, incorporating karaka (e.g., agent, theme, etc.), non-karaka (e.g. possession, purpose) and other (part of) relations. A complete tag-set of dependency relations based on CPG can be found in [7]. The ones starting with 'k' are largely Paninian karaka relations, and are assigned to the arguments of a verb.

4 Proposed Model

4.1 Vector Representations

As mentioned earlier, in order to create the semantic space, we extract a corpus of $\tilde{1}0$ GB from the web. We then expand this corpus by adding n-gram sentences for each of the

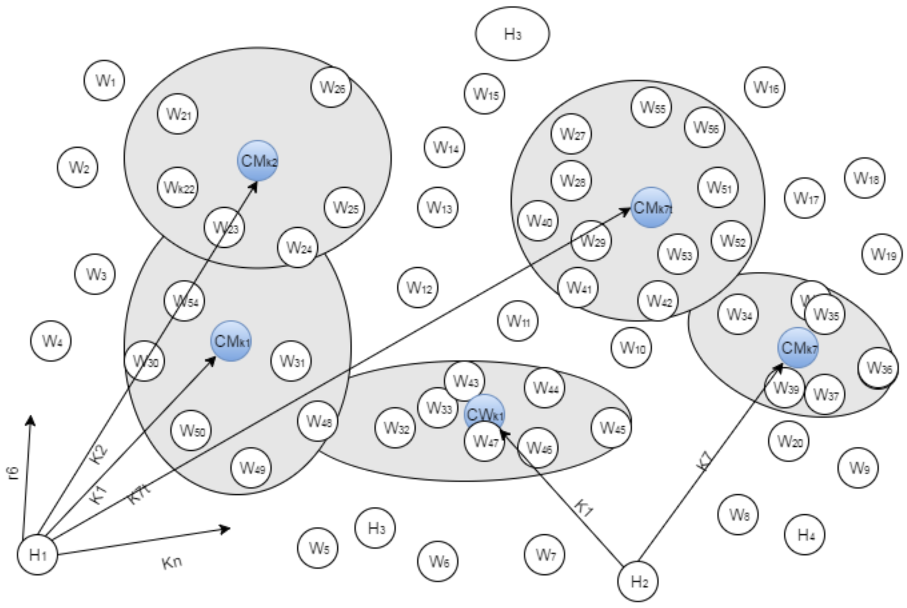

Figure 1. Working of Proposed Approach, W = word, C = Cluster center

sentences in the corpus. Given a sentence S = A B C D E F G, with each alphabet a word, the n-gram sentences for S are S_{tri} = (A_B_C) (B_C_D) (C_D_E) (E_F_G) and S_{bi} = (A_B) (B_C) (C_D) (D_E) (E_F). In both the cases the original word order is preserved.

4.2 Cluster Formation

We divide the dependency annotated treebank data (HDTB) containing 19,254 sentences into 3 parts -

- Cluster train data (16,000)
- Train data (2604)
- Test data (650)

This section discusses creating clusters from the cluster train data.

- For the purpose of this work, the word tokens we use can belong to four of different categories:
  - a tri-gram
  - a bi-gram
  - a word form
  - a lexical root of the word agglutinated with its vibhakti

For example given a sentence
S = [उसे] [गोहर बेगम भी] [कहा जाता था] [।]
S = [To Her] [Gohar Queen also] [called used to be]
S = she used to be called Queen Gohar also

- tri-gram of a chunk head if any
  Example: कहा जाता था , कहा + या जाता था - (Used to be called)
- bi-gram of a chunk head if any
  Example: कहा जाता , कहा + या जाता - (used to be Called)
- word form of a chunk head
  Example: कहा - (say)
- lexical root of a chunk head + vibhakti
  Example: कहा + या - (to be called)

– For each sentence in the cluster train data, we group together all the word tokens that have a common head token and are connected to it through the same dependency relationship resulting in clusters of edges with common heads and dependency relations

(1) मोहन ने खाना रात में खाया$_{head}$ ।
Mohan food in$_{n}$ight eat+past
Mohan ate food in the night.

(2) गणेश के भाई राम ने सेब खाया$_{head}$ ।
Ganesh's Brother Ram apple eat+past
Ganesh's Brother Ram ate an apple.

– The above two examples 1 and 2 have a common verb खाया(ate) as head with राम(Ram), मोहन}(Mohan), {खाना(to eat),सेब}(Apple), {रात}(Night) as children with k1, k2, k7t, r6 dependency relationships respectively. Therefore the clusters for खाया(ate) as a head would be

खाया(ate) -> k1 {भाई(Brother), मोहन(Mohan)} ; खाया(ate) -> k2 खाना(to eat), सेब}(Apple) ;
खाया(ate) -> k7t {रात}(Night); भाई(Brother) -> r6 {गणेश}(Ganesh) ;

– After building the aforementioned clusters for all the sentences in the cluster train data, we map each of them into the vector space of word embeddings mentioned in section 4.1.

– We now have vector representations of each of the word tokens in the clusters. we use these word vectors to calculate the mean of each of the clusters. The mean of a cluster is the sum of all the word vectors in the cluster divided by the number of words in the clusters. (CW$_{k1}$, CW$_{k2}$, CW$_{k7t}$ in Figure-1)

– The semantic distance from any word vector in the vector space to a cluster is the cosine distance between the word vector and the mean of the cluster. As we will explain in the later sections (4.4), this distance is used as a feature to determine the extent to which a word belongs to a cluster.

4.3 Population Using Hindi Wordnet

Due to the sparsity of data, the aforementioned clusters are not only few in number but are not densely populated. To be able to construct a well informed feature, we need more dense and formed clusters. To improve the cluster quality, we take advantage of the observation that two synonyms in their root form and in the right context associate with a head token with the same dependency relation.

The context however, has to be carefully identified since all the synonyms need not exhibit this behavior. To identify such contexts, we experimented with various methods like populating the clusters with all synonyms, moving the mean vector in the semantic space based on the newly added word vectors although showed improvement, was not very fruitful. While Hindi WordNet (HWN) provides synonyms of only 4 POS categories (Noun, Adjective, Verb, Adverb), populating the clusters with just the word vectors of only these categories creates discrepancies.

We also experimented populating the clusters with HWN without modifying the mean vectors of the clusters. While this worked better than the earlier approaches, we realized that there is noise in uni-gram population. Therefore we employ this approach but only populate word tokens that are bi-grams or tri-gram with their respective synonyms. The total number of clusters after HWN population came upto $\tilde{3}0,000$ uni-gram, $\tilde{2}0,000$ bi-gram and $\tilde{5},000$ tri-gram clusters.

In the below example, dependency label clusters associated with अधिगत–करने_दिया (Allowed to acquisition) have been listed. The dependency framework of the data($\bar{H}$DTB) is described in [8]

– **head** - अधिगत–करने_दिया (Allowed to acquisition)
– Cluster **rt** - राहत_के_लिए (For the sake of relief), मदद_के_लिए (For the sake of Help)
– Cluster **vmod** - मंत्री_के_खिलाफ (Against the Minister)
– Cluster **k7p** - जम्मू–कशमीर_में (In Jammu-Kashmir) , कशमीर_में (In Kashmir)
– Cluster **k1** - एसोसिएशन_ने (Association has) , आरगनाइजशन_ने (Organistaion has)
– Cluster **k7t** - शनीचर_को , शनिवार_को (On Saturday)
– Cluster **rh** - इष्ट_से , मकसद_से (With Motive)

[1] The below example shows the clusters of अधिगत–करने_दिया (Allowed to acquisition) after they have been populated using HWN.

– **head** - अधिगत–करने_दिया , खोल–करने_दिया , कोश–करने_दिया, गिलाफ–करने_दिया, आवरण–करने_दिया, ओपन–करने_दिया, उधडना–करने_दिया, निकालना–करने_दिया, खोलना–करने_दिया (Allowed to acquisition)
– Cluster **rt** - राहत_के_लिए (For the sake of relief) मदद_के_लिए(For Help) , सहा–यता_के_लिए (For Help), सहयोग_के_लिए (For Assistance), इमदाद_के_लिए (For Aid)
– Cluster **vmod** - मंत्री_के_खिलाफ (Against Minister)
– Cluster **k7p** - जम्मू–कशमीर_में (In Jammu-Kashmir), कशमीर_में (In Kashmir) , जम्मू–और–कश्मीर_में (In Jammu and Kashmir), जम्मू–एवं–कश्मीर_में (In Jammu and Kashmir), कश्मीर_में (In Kashmir)
– Cluster **k1** - एसोसिएशन_ने (Association), आरगनाइजशन_ने (Organisation)
– Cluster **k7t** - शनीचर_को , शनिवार_को (On Saturday)
– Cluster **rh** - इष्ट_से (Favoured By) , मकसद_से (With aim), नीयत_से (With Intention), आशय_से (With Intent), मतलब_से (With Meaning), लक्ष्य_से (With Goal), साध्य_से (With End)

4.4 Cluster Assignment

We apply KNN algorithm on the aforementioned clusters to determine the possible dependency relationship between a word token and its head for a given sentence. Given a word token and its possible head, the word token is assigned the cluster label of the most semantically closer one out of all of the clusters of the head word token. The semantic similary is defined as the cosine distance between the token word vector and the mean vector of the cluster. We chose to work with KNN algorithm due to its simplicity and prior sucess in other NLP applications. However, we expect that our approach can function with other clustering algorithms.

(3) मंदिर के गर्भगृह में सात घोड़ों पर सवार सूर्यदेव की प्रतिमा है ।
Of Temple in sanctuary seven horse riding of sun god there is statue
There is statue of Sun God riding seven horses in sanctuary of temple.

For the example sentence 3, गर्भगृह (sanctuary) is the governor of मंदिर (temple) with dependency label r6 connecting them. Table 1 shows the closest cosine distances between the word vector of मंदिर (temple) and some of the clusters of the head token गर्भगृह (sanctuary). From the table it can be inferred that मंदिर (temple) is closest to r6 cluster.

[1] rt - purpose, vmod - verb modifier, k7p - location in place, k1 - karta, k7t - location in time, rh - cause-effect [8]

Dep Rel	Dist to cluster
nmod	4.3755646
k2	4.9673181
r6	1.783821
rsym	4.1947953
nmod$_k$2inv	4.815473
nmod$_k$1inv	4.4694099
rsp	4.9673185
rs	4.51663868

Table 1. N-grams

5 Experiments and Results

In this section we describe the features we add to the baseline models of both the transition based and graph based models. The data we use for this consists of 3254 sentences extracted from HDTB data of which 2,242 sentences are used as train data and 891 as test data. Tables 2 and 3 show the accuracies of each of the models against a 5fold cross validated data of 2,604 train and 650 test sentences.

While composing the cluster based features, we follow a basic hierarchy of assigning significance to tri-grams followed by bi-grams and then uni-grams. If available, the tri-gram of a given head or a child or both of an arc is picked first followed by their respective bi-grams and uni-grams. Cluster based features are named after the label of the cluster they belong to. Therefore, according to Paninian framework[8], [7] a total of 40 (for each dependency label) cluster features are possible in this scenario.

5.1 Transition Based Model

Transition based parsing systems use a model parameterized over transitions, such that every transition sequence from the designated initial configuration to some terminal configuration derives a valid dependency graph. The set of training instances for the learning problem are pairs (c, t) such that t is the correct transition out of c in the transition sequence that derives the correct dependency graph for some sentence x in the training set T.

Each training instance (c, t) is represented by a feature vector f(c, t), where features are defined in terms of arbitrary properties of the configuration c, including the state of the stack, the input buffer and the partially built dependency graph. Many features involve properties of the two target tokens that could be connected by an edge, the token on top of the stack and the first token in the input buffer. The full set of features used by the base model for Hindi is described in [36]. We use an implemented version of arc eager algorithm by [37] and support vector machines as the linear model to learn transition scores.

Since the cluster features have to be added during the **inference time** when the head of a possible edge if any, between the token on top of the stack and the first token in the buffer is still undecided, it is impossible to decisively deduce cluster features. After various experiments we have found implementing both the cases, two cluster features one with each of the tokens as the head as the optimal scenario. Both are added to the baseline features in the configuration vector and appropriate feature necessary to make an accurate prediction is left for the parser to decide. These features are implicitly added through the polynomial kernel used to train the SVM. In the example 3 between the tokens गर्भगृह (sanctuary) and मंदिर (temple), the two cluster based features are 'r6', the closest cluster label मंदिर (temple) is to with गर्भगृह (sanctuary) as the head and rs the closest cluster label गर्भगृह (sanctuary) is to with मंदिर (temple) as the head.Table 2 shows the accuracies[2] of both the baseline and proposed models for the transition based parser.

	LAS	UAS	LS
Baseline Model	78.16%	86.02%	73.36%
Cluster Based Model	79.04%	86.67%	74.22%

Table 2. Transition Based Model Accuracies

5.2 Graph Based Model

Graph-based dependency parsers parameterize a model over smaller substructures in order to search the space of valid dependency graphs and produce the most likely one. The simplest parameterization is the arc-factored model that defines a real-valued score function for arcs s(i, j, l) and further defines the score of a dependency graph as the sum of the score of all the arcs it contains. The specific graph-based model studied in this work is that presented by [38], which factors scores over pairs of arcs (instead of just single arcs) and uses near exhaustive search for unlabeled parsing coupled with a separate classifier to label each arc. We use Maxent as the base line classifier with the settings suggested in [36].

Unlike Transition-based models, since the edges of the trees have already been determined we are aware of the head and the child tokens of a given edge in a tree. Hence we create only one cluster based feature to add to the feature vector to help predict the dependency labels. In the example 3 since the head is known to be गर्भगृह(sanctuary) and the child मंदिर(temple), the corresponding cluster based feature is 'r6', the closest cluster label मंदिर(temple) is to with गर्भगृह(sanctuary) as the head. While in transition-based models the cluster based features are used to help predict edges as well as dependency

[2] LAS: Labeled Accuracy Score; UAS: Unlabled Accuracy Score; LS: Labeled score

labels, in graph based models, cluster based features only help predict the labels of trees. Hence there is no improvement in Unlabelled Attachment Score(UAS).

All features are conjoined with the FEATS column of the train and test data in conll format. Table 2 shows the accuracies of both the baseline and proposed models for the transition based parser. As can be noticed from the tables 2 , 3 there is more improvement in accuracies for graph based models compared to transition based models. This could be due to disambiguity in head and children in the former case.

	LAS	UAS	LS
Baseline Model	78.78%	86.82%	78.78%
Cluster Based Model	79.81%	86.82%	79.81%

Table 3. Graph Based Model Accuracies

6 Error Analysis

In this section we discuss the linguistic errors caused by the base line models that are being disambiguating succesfully by our approach.

(4)	वो	आदमी	भूक	से	मर	गया
	That	man	hunger	of	die	did

That man died of hunger

(5)	उसने	मुझे	पत्थर	से	मार
	He	me	rock	with	hit

He hit me with a rock

In the above examples more context is needed to diambiguate the role of the post-position से(with) .In examples 4 it is a cause while in example 5 it is an instrument. Due to the bigram clustering in our approach, भूक_से (with hunger) and पत्थर_से (with a rock) are being accurately labeled.

(6)	बेटा	बहू	की	खुशी	के लिये	अब	यह	भी	करना
	Son	daughter-in-law's	for	happiness	for	now	this	also	to-be-done
	पड़ेगा	।							
	has	.							

`Son for daughter-in-law's happiness now this also has to be done .'

बहू (daughter-in-law) is the genitive of खुशी (hunger) and is being identified accurately while earlier it was confused for the subject. similarly के लिये (for the sake of) in the trigram खुशी के लिये(for the sake of happiness) is helping identify खुशी(happiness) as the 'rt'(reason) relation of करना(to do)

(7) रजनी रोहित को झूलाघर मेंछोड़कर देर स्कूल पहुंचती ।
Rajni Rohit in park after leaving late school reached
`Rajni after leaving Rohit in the park reached school late'

झूलाघर(park) is being accurately identified as k7p(location in place) to छोड़कर(let go) and देर as k7t(location in time) to पहुंचती(reached) while the base models are getting them wrong.

(8) स्कूल की दूसरी शिक्षिकाओं से पूछताछ की पर सन्तुष्ट नहीं
Of school other teachers inquiring even after satisfied didn't get
हुई ।

`Didn't get satisfied even after inquiring other teachers of school .'

The bigram स्कूल की is helping to identify as r6 to शिक्षिकाओं where as शिक्षिकाओं से is being identifed as k2 to की. similarly the cases where पर is the nominal post-position, there is an ambiguity between a place and time. Our bigram models seem to fix most of these issues.

7 Conclusion and Future Work

In this paper, we have presented a simple but effective semi-supervised learning approach and demonstrated that it achieves substantial improvement over a competitive baseline in two broad-coverage dependency parsing tasks. This work could be improved with better development of the clustering algorithm to reflect the syntactic behavior of words; e.g., an algorithm that attempts to maximize the likelihood of a treebank, according to a probabilistic dependency model. Alternately, one could design clustering algorithms that cluster entire head-modifier arcs rather than individual words. Another idea would be to integrate the clustering algorithm into the training algorithm in a limited fashion. For example, after training an initial parser, one could parse a large amount of unlabeled text and use those parses to improve the quality of the clusters. These improved clusters can then be used to retrain an improved parser. In the future, we would also extend this approach for other languages like English, Telugu etc.

References

1. Kingsbury, P., Palmer, M., Marcus, M.: Adding semantic annotation to the penn treebank. In: Proceedings of the human language technology conference, Citeseer (2002) 252--256

2. Fujita, S., Bond, F., Oepen, S., Tanaka, T.: Exploiting semantic information for hpsg parse selection. Research on language and computation **8** (2010) 1--22
3. Huang, F., Ahuja, A., Downey, D., Yang, Y., Guo, Y., Yates, A.: Learning representations for weakly supervised natural language processing tasks. Computational Linguistics **40** (2014) 85--120
4. Koo, T., Carreras Pérez, X., Collins, M.: Simple semi-supervised dependency parsing. In: 46th Annual Meeting of the Association for Computational Linguistics. (2008) 595--603
5. Brown, P.F., Desouza, P.V., Mercer, R.L., Pietra, V.J.D., Lai, J.C.: Class-based n-gram models of natural language. Computational linguistics **18** (1992) 467--479
6. Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C.: A neural probabilistic language model. journal of machine learning research **3** (2003) 1137--1155
7. Bharati, A., Sharma, D.M., Husain, S., Bai, L., Begam, R., Sangal, R.: Anncorra: Treebanks for indian languages, guidelines for annotating hindi treebank (2009)
8. Begum, R., Husain, S., Dhwaj, A., Sharma, D.M., Bai, L., Sangal, R.: Dependency annotation scheme for indian languages. In: IJCNLP, Citeseer (2008) 721--726
9. Miller, S., Guinness, J., Zamanian, A.: Name tagging with word clusters and discriminative training. In: HLT-NAACL. Volume 4. (2004) 337--342
10. Ritter, A., Clark, S., Mausam, Etzioni, O.: Named entity recognition in tweets: An experimental study. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. EMNLP '11, Stroudsburg, PA, USA, Association for Computational Linguistics (2011) 1524--1534
11. Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N., Smith, N.A.: Improved part-of-speech tagging for online conversational text with word clusters, Association for Computational Linguistics (2013)
12. Ratinov, L., Roth, D.: Design challenges and misconceptions in named entity recognition. In: Proceedings of the Thirteenth Conference on Computational Natural Language Learning, Association for Computational Linguistics (2009) 147--155
13. Grave, E., Obozinski, G., Bach, F.: Hidden markov tree models for semantic class induction. In: CoNLL-Seventeenth Conference on Computational Natural Language Learning. (2013)
14. Spitkovsky, V.I., Alshawi, H., Jurafsky, D.: Lateen em: Unsupervised training with multiple objectives, applied to dependency grammar induction. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics (2011) 1269--1280
15. Finkel, J.R., Kleeman, A., Manning, C.D.: Efficient, feature-based, conditional random field parsing. In: ACL. Volume 46. (2008) 959--967
16. Tratz, S., Hovy, E.: A fast, accurate, non-projective, semantically-enriched parser. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics (2011) 1257--1268
17. Hisamoto, S., Duh, K., Matsumoto, Y.: An empirical investigation of word representations for parsing the web. In: Proceedings of ANLP. (2013) 188--193
18. Øvrelid, L., Nivre, J.: When word order and part-of-speech tags are not enough--swedish dependency parsing with rich linguistic features. In: Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP). (2007) 447--451
19. Bharati, A., Husain, S., Sharma, D.M., Sangal, R.: A two-stage constraint based dependency parser for free word order languages. In: Proceedings of the COLIPS International Conference on Asian Language Processing 2008 (IALP). (2008)
20. Ambati, B.R., Gadde, P., Jindal, K.: Experiments in indian language dependency parsing. Proceedings of the ICON09 NLP Tools Contest: Indian Language Dependency Parsing (2009) 32--37

21. Agirre, E., Bengoetxea, K., Gojenola, K., Nivre, J.: Improving dependency parsing with semantic classes. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2, Association for Computational Linguistics (2011) 699--703
22. Jain, S., Jain, N., Tammewar, A., Bhat, R.A., Sharma, D.M.: Exploring semantic information in hindi wordnet for hindi dependency parsing. In: IJCNLP. (2013) 189--197
23. Tammewar, A., Singla, K., Agrawal, B., Bhat, R., Sharma, D.M.: Can distributed word embeddings be an alternative to costly linguistic features: A study on parsing hindi. In: Proceedings of the 6th Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2015). (2015) 21--30
24. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. (2013) 3111--3119
25. Goldberg, Y., Levy, O.: word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. arXiv preprint arXiv:1402.3722 (2014)
26. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: EMNLP. Volume 14. (2014) 1532--1543
27. Turney, P.D., Pantel, P., et al.: From frequency to meaning: Vector space models of semantics. Journal of artificial intelligence research **37** (2010) 141--188
28. Baroni, M., Lenci, A.: Distributional memory: A general framework for corpus-based semantics. Computational Linguistics **36** (2010) 673--721
29. Bullinaria, J.A., Levy, J.P.: Extracting semantic representations from word co-occurrence statistics: A computational study. Behavior research methods **39** (2007) 510--526
30. Turian, J., Ratinov, L., Bengio, Y.: Word representations: a simple and general method for semi-supervised learning. In: Proceedings of the 48th annual meeting of the association for computational linguistics, Association for Computational Linguistics (2010) 384--394
31. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. Journal of Machine Learning Research **12** (2011) 2493--2537
32. Fellbaum, C.: A semantic network of english: the mother of all wordnets. In: EuroWordNet: A multilingual database with lexical semantic networks. Springer (1998) 137--148
33. Bhattacharyya, P.: Indowordnet. In: In Proc. of LREC-10, Citeseer (2010)
34. Bhatt, R., Narasimhan, B., Palmer, M., Rambow, O., Sharma, D.M., Xia, F.: A multi-representational and multi-layered treebank for hindi/urdu. In: Proceedings of the Third Linguistic Annotation Workshop, Association for Computational Linguistics (2009) 186--189
35. Palmer, M., Bhatt, R., Narasimhan, B., Rambow, O., Sharma, D.M., Xia, F.: Hindi syntax: Annotating dependency, lexical predicate-argument structure, and phrase structure. In: The 7th International Conference on Natural Language Processing. (2009) 14--17
36. Husain, S., Mannem, P., Ambati, B., Gadde, P.: The icon-2010 tools contest on indian language dependency parsing. Proceedings of ICON-2010 Tools Contest on Indian Language Dependency Parsing, ICON **10** (2010) 1--8
37. Bhat, R.A., Bhat, I.A., Sharam, D.M.: Improving dependency parsing of hindi and urdu by modeling syntactically relevant phenomena. ACM Transactions on Asian and Low-Resource Language Information Processing (under review) (2016)
38. McDonald, R., Crammer, K., Pereira, F.: Online large-margin training of dependency parsers. In: Proceedings of the 43rd annual meeting on association for computational linguistics, Association for Computational Linguistics (2005) 91--98