A Graph Based Method for Building Multilingual Weakly Supervised Dependency Parsers

Jagadeesh Gorla, Anil Kumar Singh, Rajeev Sangal, Karthik Gali, Samar Husain, and Sriram Venkatapathy

Language Technologies Research Centre, IIIT, Hyderabad, India
{jagadeesh,anil,samar,sriram}@research.iiit.ac.in
{sangal,karthikg}@students.iiit.ac.in

Abstract. The structure of a sentence can be seen as a spanning tree in a linguistically augmented graph of syntactic nodes. This paper presents an approach for unlabeled dependency parsing based on this view. The first step involves marking the chunks and the chunk heads of a given sentence and then identifying the intra-chunk dependency relations. The second step involves learning to identify the inter-chunk dependency relations. For this, we use an initialization technique based on a measure we call Normalized Conditional Mutual Information (NCMI), in addition to a few linguistic constraints. We present the results for Hindi. We have achieved a precision of 80.83% for sentences of size less than 10 words and 66.71% overall. This is significantly better than the baseline in which random initialization is used.

Keywords: Weakly Supervised Learning, Dependency Parsing, Multilingual Processing, South Asian Languages, Association Measures.

1 Introduction

Parsing a sentence can be described as finding the correct syntactic structure of that sentence according to a particular formalism. Most of the work on parsing so far can be categorized as either based on rules or based on supervised learning [8,24,6,4,1]. Fairly good parsers are available for English [3] and for some other languages [7]. Some of the latest work was presented at the CoNLL Shared Task Session of EMNLP-CoNLL 2007 [17]. There has also been work on learning models of dependency trees [16,22,21,23,25,15]. However, many other languages of the world still lack good parsers. This is mainly because these languages do not have the resources required for building either rule based parsers (extensive computational grammars) or supervised parsers (treebanks). Since the researchers working on most of these languages usually happen to be short of funding and other support, we need to find reasonably good methods for unsupervised or weakly supervised parsing, either for direct use or for making the task of creating treebank like resources easier.

A sentence can be seen as a graph consisting of syntactic units as the nodes and dependency relations as the edges. The weight of an edge represents the degree of association between the two nodes. We can view the syntactic structure of the sentence as the best spanning tree for that graph. There has been some previous work where the problem of parsing was modeled as finding the maximum spanning tree (MST) in a graph representing the sentence [14,13]. Our work is in a similar direction. We also represent the sentence as a graph, but we have 'chunks' rather than words as the nodes. This is because we are focusing on languages which have fixed word order inside chunks, but which can have relatively free order as far as chunks are concerned. One example of such languages are the languages of the South Asian linguistic area [10]. Having chunks as the nodes reduces the complexity of the problem significantly and is likely to increase the accuracy. Another justification for doing this is that the intra-chunk dependencies in these languages are very easy to find by using some simple rules, as we will explain later. It is also relatively quite easy to identify the chunks using a few rules defined in terms of the part of speech (POS) tags. It should be noted that the notion of a chunk as used by us is somewhat specific to our purposes, i.e., identifying dependency relations. So, for our purposes, a chunk is a sequence of words inside which the order of words is fixed and the dependencies are very easily identifiable. This is why we do not make the assumption that a chunker is available for the languages concerned, as we can identify the 'chunks' using some simple rules.

One assumption that we do make is that a POS tagger is available for the concerned language. The parsing algorithm runs over the sequence of POS tags for the given sentence. This assumption is valid for several South Asian languages [19,20], even though there is still a lot of scope for improving the accuracy of the POS taggers. For learning dependency relations, we can either use manually POS tagged data (if available), or we can use the output of the POS tagger on a raw corpus. In our experiments, we do the latter. Due to this and the other reasons mentioned above, our method can be easily extended for other similar languages, even though we have experimented only on Hindi so far.

During the last few years there has been a steady progress in the area of unsupervised parsing [2,12], but most of the work is based on phrase structure grammars, rather than dependency grammars. Very few efforts have been made towards building unsupervised or weakly supervised dependency parsers. Klein and Manning had [12] proposed a hybrid approach, which combines constituency and dependency models. This approach yielded 77.6% f-score on WSJ-10 corpus.

The model learnt using the method described in this paper can also be used as Dependency Language Model [11]. This could be an important application of the work even if the parser, as it is, may not be directly usable for some practical applications.

2 Overview of the Parsing Method

In the training phase, our method requires a raw text corpus and a POS tagger. The corpus is first POS tagged. After that, we identify the chunks, chunk heads and the intra-chunk dependencies using simple rules (see Section-3). From then on, we basically work with sequences of POS tags of chunk heads.

To create the parsing (training) model, we have designed a non-projective weakly supervised dependency parsing algorithm to learn the dependency relations between the head words of chunks using an Expectation-Maximization (EM)-like iterative algorithm. The approach that we describe in this paper to create the initial dependency

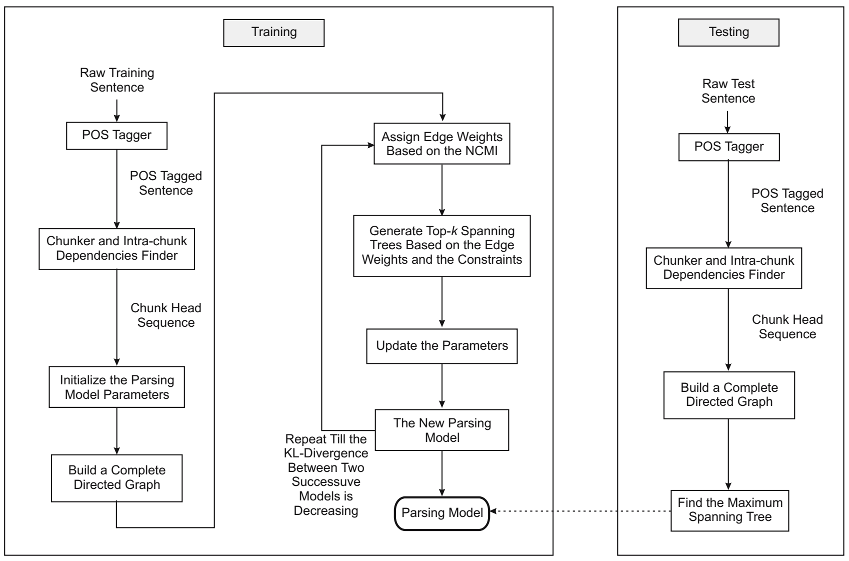
Training
Raw Training Sentence
POS Tagger
POS Tagged Sentence
Chunker and Intra-chunk Dependencies Finder
Chunk Head Sequence
Initialize the Parsing Model Parameters
Build a Complete Directed Graph
Assign Edge Weights Based on the NCMI
Generate Top-k Spanning Trees Based on the Edge Weights and the Constraints
Update the Parameters
The New Parsing Model
Repeat Till the KL-Divergence Between Two Successuve Models is Decreasing
Parsing Model
Testing
Raw Test Sentence
POS Tagger
POS Tagged Sentence
Chunker and Intra-chunk Dependencies Finder
Chunk Head Sequence
Build a Complete Directed Graph
Find the Maximum Spanning Tree

Fig. 1. Overview of the parsing method based on syntactic structure as the best spanning tree in a linguistically augmented graph of syntactic nodes

model uses a measure called Normalized Conditional Mutual Information (NCMI) along with a few linguistic constraint which are applicable across all the major South Asian languages.

In this proposed approach to build a parser, no treebank is required and it can also be applied to other free order languages by changing the linguistic constraints, which are very few in number. All the rules and the model parameters of parsing model in this approach are based on only POS tags.

One of the reasons for using a sequence of POS tags for learning is that the dependency relations between two words mostly depend on their POS tags, since words of the same POS tags are usually mutually substitutable. This is an established practice and was used by Klein [12]. Also, as we are using some rules to identify the chunks and chunk heads, it is easy to define such rules on tags instead of lexical items.

Figure-1 illustrates the basic ideas used in the method described in this paper.

3 Syntactic Properties of the Languages Covered

As indicated earlier, our method is suitable for languages with certain properties. These properties include a relatively free word order nature. ‘Relatively’ because the order within a chunk may be fixed and languages like Hindi are verb final languages. A sentence in such languages is typically divided into non-overlapping phrases (chunks), and these chunks can occur in any order without affecting the core meaning of the sentence. Each chunk consists of one content word, referred to as the *head word* H, and several other words, which are function words F. The head of the chunk is the main element of the chunk and it carries the meaning of the chunk and can occur independently. For example, consider the following Hindi sentences:

S1: *[merI nayI kiwAba] [rAma] [paDZa rahA hE].*
[my new book] [Ram] [is reading]
S2: *[rAma] [merI nayI kiwAba] [paDZa rahA hE].*
[Ram] [my new book] [is reading]

Both the sentences S1 and S2 convey the same meaning, even though *[merI nayI kiwAba]* and *[rAma]* are swapped in S2. Here *[merI nayI kiwAba]* and *[rAma]* are noun chunks and *[paDZa rahA hE]* is a verb chunk. The chunks are (internally) fixed word order units, but they can be combined in almost any order. The underlined word in each chunk is the head word and the other words are function words.

One important condition while building a dependency parser for South Asian languages is that we can not apply the projectivity constraint because these languages are relatively free word order languages. Our study of the dependency relations inside the chunks clearly shows that, almost always, the words inside the chunk modify the head of that chunk.

In our experiments, we have used some rules which generalize to many South Asian languages. Some of these rules used by the Chunker and Intra-chunk Dependency Finder (CIDF) are given below[1]:

1. r1: (QFNUM | QF | INTF | QFN | JJ)* (NN | (NNPC)* NNP | (NNC)* NN | PRP) (PREP | NLOC | RP | SYM)*
2. r2: (NEG)* (VRB | NVB | VJJ | VFM | VAUX) + (PREP | NLOC | RP | SYM)*
3. r3: (RB)+
4. r4: (CC)+
5. r5: (JJ)+

The rule $r1$ is used for identifying noun chunks in a sentence, $r2$ is for verb chunks, $r3$ is for adverbs, $r4$ is for conjunctions, and $r5$ is for adjectives. These rules are applied one after the other (i.e., in the order given above) to identify the different chunks in the given sentence.

We have conducted some experiments to evaluate the performance of the CIDF. The precision for chunking is 77%, for intra-chunk dependencies it is 96%, and for finding the head of the chunk it is 98%. There are some issues still to be resolved, e.g. if PRP NN is a tag sequence then it is difficult to decide whether to combine them as a chunk or not. If the PRP is a demonstrative PRP then it will be a chunk, otherwise there will be two different chunks. This situation is not currently handled by the CIDF.

4 Weakly Supervised Dependency Parsing Model

In this section, we present an approach to create an weakly supervised dependency parsing model using chunks head sequences. It captures the strength of a dependency relation between any two chunk head tags at a particular distance.

[1] We are using a POS tagset which has been designed for many Indian languages. The details are available at http://shiva.iiit.net/SPSAL2007/downloads.php

4.1 Notation

We view the sentences of a language L as sequences of word tokens drawn from some set of word types or vocabulary of L. Let $V = \{t_0,\ t_1, \cdots t_v\}$ be the vocabulary of the language L and $S =< x_0,\ x_1, \cdots, x_n >$ be a sentence of n words such that $x_0 = ROOT$, $\forall n$ such that $x_n \in V$. Let $C = \{S_0,\ S_1, \cdots S_N\}$ be the set of N sentences in the language L. Let $G_S = (V_S, E_S)$ be a complete directed graph of a sentence S such that,

1. the set of vertices $V_S = \{x_0,\ x_1, \cdots, x_n\}$
2. the set of edges $E_S = \{< x,\ y >_S | \forall x, y \in V_S\}$

where G_S is a graph such that each word in the sentence is a node and there is a directed edge between every pair of nodes, corresponding to the dependencies. By definition, G_S is a digraph. G_S encodes all possible dependencies among the words (actually, chunk heads) of the sentence S. Thus, every possible dependency graph of S must be a subgraph of G_S. Let $< x,\ y >_S$ represent the edge from x to y in the sentence S.

Let $x \rightarrow^+ y$ be a dependency relation that is true if and only if there is a non-empty directed path from node x to node y in some graph under consideration. A directed spanning tree of a graph G_S that originates out of a particular node $x_r \in V_S$, is any subgraph $T = (V_T, E_T)$ such that,

1. $V_T = V_S$ and $E_T \subseteq E_S$
2. $\forall x_j \in V_T,\ x_r \rightarrow^+ x_j$ if and only if $x_r \neq x_j$
3. If $< x_i,\ x_j >_S \ \in E_T$, then:
   $< x_m,\ x_j >_S \ \notin E_T \ \forall x_m \neq x_i$

Let $T(G_S)$ be the set of all directed spanning trees of the graph G_S. As McDonald et al. [14] noted, there is a one-to-one correspondence between spanning trees of G_S and dependency graphs of S. This implies that $T(G_S)$ is the set of all possible projective and non-projective dependency graphs for the sentence S.

For a given sentence S, using the parsing model, we can estimate the conditional probability $P(T|S)$ and the parser finds the most likely parse of the sentence using:

$$T_{best} = \arg\max_{T} P(T|S) \tag{1}$$

We assume that each dependency decision is independent. The class of dependency models which follow this assumption are called **edge-factored models** [18,14]. Under this assumption, every edge in G_S of a sentence S is associated with a score $Score(x, y) \geq 0$ that maps edge between x and y to a real valued score ranging from 0 to 1. These scores represent the likelihood of the dependency relation occurring from word x to y. We refer to this score as the dependency score of x being the head of y and we denote it by $Score(x, y)$. The way $Score(x,\ y)$ is calculated depends on the framework in which it is being used. For example, in a generative probabilistic model such as Paskin's [18] it could represent the conditional probability of x being generated by y.

Based on the above assumption, we define $P(T|S)$ as follows:

$$P(T|S) = \prod_{(x,\ y) \in E_T} Score(x,\ y) \tag{2}$$

We can also write T_{best} of a sentence S as follows:

$$T_{best} = \arg\max_{T \in T(G_S)} P(T|S) = \arg\max_{T \in T(G_S)} \prod_{(x,y) \in E_T} Score(x,\ y)$$

Here, $Score(x,\ y)$ represents the probability of x being the head of y. It can also be represented as $P(x|y)$. McDonald et al.[14] showed that this can be solved in $O(n^2)$ time using the Chu-Liu-Edmonds algorithm for standard digraphs [5,9]. We will use the parsing algorithm proposed by McDonald et al.[14] to parse a given test sentence, once the training has been completed.

We need to estimate the parsing model parameters, i.e., $P(x|y)$ to find T_{best} for a sentence. Before describing the way $Score(x,\ y)$ (or $P(x|y)$) is estimated, we define the following:

1. We denote the sum of the scores of all the possible output parses of a given input sentence S as Z_S:

$$Z_S = \sum_{T \in T(G_S)} P(T|S) = \sum_{T \in T(G_S)} \prod_{(x,\ y) \in E_T} P(x|y)$$

where $x, y \in V_S$

2. The expected value (or the dependency score) of each edge in G_S for a sentence S is represented as $< P(x|y) >_S$ and is computed as:

$$< P(x|y) >_S = \sum_{T \in T(G_S)} P(T|S) \times I(< x,\ y >,\ T)$$

where $I(< x, y >,\ T)$ is an indicator function that is equal to 1 when the edge $< x,\ y >$ is in the tree T and is zero otherwise.

An estimate based on the identities of the two tokens alone is problematic and the lexical distance between the words will strongly influence the likelihood of one word modifying other [6]. So, we include the distance while deciding whether two words are related or not. We estimate, $< P(x|y, d) >_S$, i.e., the probability of x being the head of y at a lexical distance d, instead of just $P < (x|y) >_S$.

4.2 Language Model for Parameter Estimation

In principle, a language model recovers the probability of a sentence $P(S)$ over all possible T given S by estimating the joint probability $P(S, T)$:

$$P(S) = \sum_T P(S, T) \tag{3}$$

In practice, we approximate $P(S)$ to the sum of tree scores (probabilities) of k best probable trees generated for the sentence S. Let $T_k(S)$ be the set of k best probable trees generated for the sentence. Each tree $T \in T_k(S)$ is the most probable tree for sentence S when each word of the sentence is taken as the root node.

$$P(S) = \sum_{T} P(S,T) \approx \sum_{T \in T(T_k(S))} P(S,T) = \sum_{T \in T(T_k(S))} \prod_{(x,y,d) \in E_T} < P(x|y,d) >_S$$

To generate the trees in $T_k(S)$, we first generate a single child, $x_r \in V_S$, of $ROOT$ and then we select the single best incoming edge of each node x_g such that $x_g \in V_S$, $x_g \neq x_r$ and T_{x_r} should satisfy the spanning tree properties (no cycles). In other words, T_{x_r} is the directed maximum spanning tree of G_S generated from the node x_r. In this way, in each tree, we have to choose the head of each word (incoming edge) in the sentence S based on the probability of each edge, i.e., $P(x|\ y, d)$, where $x,\ y \in V_S$.

It is very unlikely that the same sentence will appear in the training data and the test data. We thus approximate $< P(x|\ y,\ d) >_S$ by $P(x|\ y,\ d)$ and estimate the dependency probability from the training sentences, where $x,\ y \in V$. From now onwards, we denote $P(x|\ y,\ d)$ as $P^d_{x,\ y}$.

Let $t_a,\ t_b \in V$. We can estimate the maximum likelihood $P^d_{t_a,\ t_b}$ by maximizing the log-likelihood $\sum_{c=1}^{N} log(P(S_c))$ subjected to the normalization constraints:

$$\sum_{t_a} P^d_{t_a,\ t_b} \ = \ 1 \tag{4}$$

and

$$P^d_{t_a,\ t_b} \geq 0 \tag{5}$$

Let x_{ci} be the i^{th} word of S_c. By solving the above constraint optimization problem with the usual Lagrange multipliers method, we get the probability $P^d_{t_a t_b}$ as:

$$P^d_{t_a t_b} = \frac{\sum_{c=1}^{N} \frac{1}{Z_{S_c}} \sum_{\substack{x_{ci}=t_a \\ x_{cj}=t_b}} < P^d_{x_{ci},x_{cj}} >_{S_c}}{\sum_{c=1}^{N} \frac{1}{Z_{S_c}} \sum_{t_a,d} \sum_{\substack{x_{ci}=t_a \\ x_{cj}=t_b}} < P^d_{x_{ci},x_{cj}} >_{S_c}} \tag{6}$$

$P^d_{t_a,\ t_b}$ are the maximum likelihood parameters for the dependency parsing model. We estimate these parameters from the head tagged corpora using an EM-like iterative algorithm described below.

Learning the Dependency Parsing Model. To create a dependency model, we first create the chunk head corpus by running a tagger on a corpus (21857 sentences) followed by running the CIDF on the tagged corpora, which gives the head corpus. We then apply the following three steps to create the parsing model:

1. Initialize the parsing model parameters
2. For each sentence in the chunk head corpus:
   (a) Construct a complete weighted directed graph with the nodes as chunk heads and the edge weights as the parsing model weights between the chunk heads
   (b) Find out what is the possible root of the sentence, i.e., is it CC or VFM

(c) Find out the possible parse trees of the sentences by finding the MST of the graph by taking the each possible root as the head of the sentence, i.e., take the possible root as the child of the dummy root and generate the MST.

3. Estimate $P^{d}_{t_a,\ t_b}$ from the set of generated trees and update the weight of each parameter and create a new model
4. Repeat from the step-2(b) with newly estimated edge weights in step-3 until the KL-divergence between the two successive models is decreasing

5 Creating Initial Parsing Model

The effectiveness of the dependency model is highly dependent on the initial dependency model weights. We initialize the dependency model using a measure called Normalized Conditional Mutual Information (NCMI) and a few linguistic constraints. NCMI is a measure of association between distant word (or POS tag) pairs, i.e., two words occurring at a particular distance d. We use NCMI between two tags as an initial undirected dependency score between them.

First we define a measure called Conditional Mutual Information (CMI), which is a modified version of the mutual information measure that takes into account the extra variable of the lexical distance between the words. The condition in the 'conditional' is the value of the distance. CMI is calculated as:

$$CMI(x, y, d) = p(x, y, d) log \frac{p(x, y, d)}{p(x)p(y)} \tag{7}$$

where x, y are the POS tags and d is the distance between x and y.

All CMI scores are then normalized to get the $NCMI$ ($NCMI \in [0, 1]$), which is calculated as:

$$NCMI = \frac{CMI - min(CMI)}{max(CMI) - min(CMI)} \tag{8}$$

As mentioned earlier, $NCMI$ gives the measure of association or interdependency score between the two tags at a particular distance. It does not represent the exact measure of dependency between two tags, but is an approximation to the measure of dependency which can be effectively used for initialization.

5.1 Dependency Constraints

Based on the linguistic reality and the frequencies of occurrence of dependency relations, we have defined a small set of constraints that any valid dependency tree must satisfy.

A tree is considered a valid dependency tree of sentence iff:

1. It has either a main verb (VFM) or a conjunction (CC) as the root of the dependency tree
2. It does not have any adjective (JJ) as the head of any verb (VFM, VJJ, VRB)

3. It does not have any noun (NN) as a child of a pronoun (PRP)
4. It does not have a dependency relation between two conjunctions (CCs) or two main verbs (VFMs).
5. If there is an adverb (RB) or a non-finite adverbial (VRB) then it modifies (i.e., is the child of) its closest main verb (VFM)
6. Two nouns (NN) are not related at a lexical distance greater than 2.

The first three constraints are based on the linguistic reality while the rest are based on frequencies.

5.2 Generating Initial Dependency Trees

This section describes the method to generate the initial set of dependency trees for each sentence using the NCMI scores and the dependency constraints and then creating the initial model from the trees. We know that the NCMI score is a rough measure of the strength of a dependency relation between two tags. From the constraints given above, we also know that two conjunctions (CC) or two main verbs (VFM) or two nouns (NN) at a lexical distance of more than 2 are not related. This means that we can modify the NCMI scores between two conjunctions, main verbs or nouns at lexical distance more than 2 to 0 (weak dependency score). And so on for other constraints. The modified NCMI scores are used to generate the most likely dependency trees for a given sentence.

Let S be the sentence and g_s be the weighted undirected complete graph of S constructed using the words/tags of the sentence as nodes. Following are the steps to generate the initial dependency trees for a sentence S:

1. Construct g_s by taking words/tags of S as nodes and the $NCMI$ score as edge weight.
2. Modify the weights of the edges connecting the node tag VNN or VRB or RB in g_s to its closest [2] VFM node in the graph to 1 (strong dependency score).
3. Generate the set of M maximum spanning trees of g_s.
   Each spanning tree of g_s represents the possible undirected dependency tree of the sentence S and the tree score (product of edge scores) represents the likelihood of tree being the undirected dependency tree of S.
4. Generate the set of directed trees from each undirected tree by taking each possible head (CC and VFM) as a root and converting the undirected tree to a directed tree by assigning directions to each edge as going out from the parent towards the child.

After applying the above steps on the sentences, we get the set of most likely parse trees of the sentences and their scores ($P(S,\ T)$). The basic intuition here is that once we modify the edge weights using the constraints, when generating the top M-spanning trees, edges with strong dependency score are likely to be included in the top MSTs. Once the initial possible trees are generated for each sentence, we can estimate the initial parsing model parameters by calculating $P^d_{t_a,\ t_b}$ as described in the previous section. One important observation here is that we are estimating the parsing model parameters

[2] Here closeness is in terms of the lexical distance between the nodes in the sentence.

using the trees generated based on the NCMI scores modified by the linguistic constraints and these linguistic constraints are used only in estimating the initial parameter values.

5.3 Parsing

The output of the training model (or learning model) described in the above sections are the dependency relations between tags at a particular distance with their corresponding probabilities. To parse a new (test) sentence, we apply the following steps:

1. Run the POS tagged and the CIDF and find chunks, chunk heads and intra-chunk dependency relations
2. Construct a complete weighted graph of the sentence by taking the chunk heads as nodes and assigning the parsing model weights for the tag-tag-distance triples to the edges
3. Compute the Maximum Spanning Tree of the sentence graph using the Chu-Liu-Edmonds algorithm [5,9]

The output MST gives us the most likely parse of the sentence.

6 Experimental Setup and Results

As not much parsed data is available for Indian Languages, we used raw (unannotated) text to build a parser and tested the parser accuracy on 1997 human tagged sentences (these sentences were not the part of the training corpus). For training, we tagged 22187 sentences using a POS tagger for which the reported performance is 88.8%. We calculated the NCMI scores between the tags at a maximum lexical distance d of 20. We used the standard evaluation metric used in supervised parsing techniques to evaluate the parser performance, i.e.:

$$Accuracy = \frac{correct\ dependency\ relations}{total\ relations}$$

We conducted two main experiments. The first was on sentences of length less than ten words and the second was on sentences of all lengths. We calculated the performance for both the sets of sentences. The results are shown in **Table-1** and **Table-2**. Note that these results are only for the accuracy of inter-chunk head dependency relations. The precision of the CIDF for intra-chunk dependency identification was found to be 88.69%, 83.21% and 90.50% for Hindi, Telugu and Bengali, respectively. Since the CIDF is able to find intra-chunk relations much more accurately, the overall performance of the parser will be greater than shown in **Table-1**, even more so because the average length of sentence in the training data was 20.34, whereas it was only 10.14 after the chunking step.

We also conducted an experiment where the initialization was performed with random values. The performance is this case (for sentences of all lengths) was 48.44%. The performance when initialization was performed with the NCMI scores modified by the linguistic constraints was 66.71%. This clearly shows that our method of initialization is making a significant difference to the performance.

Table 1. Parser Evaluation: Sentence Length and Initialization Method

Sentence Length	Accuracy
≤ 10	80.83%
Overall	66.71%

Initialization	Accuracy
Random (Baseline)	48.44%
Only NCMI	64.13%
NCMI + Constraints	66.71%

7 Future Work

There are several possible areas of further research as an extension of this work. One of them is improving the learning and parsing algorithms. Right now, the parser is using only the tags to learn the dependency relations. Since we know that lexical information can be crucial in parsing, we can use such information, especially the post-positions or case markers, to make the parser more accurate. Such improvements can perhaps make the parser practically usable. One other very important area for future work is domain adaptation of the parser for restricted domains as has been successfully attempted for many other languages [17].

8 Conclusion

In this paper we presented a method for weakly supervised dependency parsing. This method is based on the idea that the problem of parsing can be defined as the computation of the best spanning tree in the complete graph generated from the syntactic units of the sentence, where these units are the nodes in the graph. In our case, we select chunks as the units because our focus was on South Asian languages which have fixed word order but free chunk order. Also, it is very easy to identify the chunks relevant for our purposes as well as to identify the intra-chunk dependencies. We described an algorithm for learning the parsing model from POS tagged sequences prepared from the unannotated training data. A novel way was used to initialize the parameters of the model (i.e., weights of the edges between two tags at particular distances). This method of initialization used a measure of association called Normalized Conditional Mutual Information (NCMI) and application of a few simple linguistic constraints. Our parser achieved an accuracy of 80.80% for sentence of length less than ten and 66.71% for sentences of all lengths. We were also able to show that our method of initialization significantly increased the performance of the parser over the baseline of random initialization.

References

1. Bharati, A., Sangal, R.: Parsing free word order languages using the paninian framework. In: Proceedings of Annual Meeting of Association for Computational Linguistics, pp. 105–111 (1993)
2. Bod, R.: An all-subtrees approach to unsupervised parsing. In: Proceedings of COLING-ACL (2006)

3. Charniak, E.: A maximum-entropy-inspired parser. In: Proceedings of the Annual Meeting of the North American Chapter of the Association for Computational Linguistics (ACL) (2000)
4. Charniak, E., Johnson, M.: Coarse-to-fine n-best parsing and maxent discriminative reranking. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL) (2005)
5. Chu, Y., Liu, T.: On the shortest arborescence of a directed graph. Science Sinica 14, 1396–1400 (1965)
6. Collins, M.: Head-driven statistical models for natural language parsing. PhD thesis, University of Pennsylvania (1999)
7. Collins, M., Hajic, J., Brill, E., Ramshaw, L., Tillmann, C.: A statistical parser for czech. In: Proceedings of the 37th Meeting of the Association for Computational Linguistics (ACL), pp. 505–512 (1999)
8. Doran, C., Egedi, D., Hockey, B.A., Srinivas, B., Zaidel, M.: Xtag system: a wide coverage grammar for english. In: Proceedings of the 15th conference on Computational linguistics, Morristown, NJ, USA, pp. 922–928. Association for Computational Linguistics (1994)
9. Edmonds, J.: Optimum branchings. Journal of Research of the National Bureau of Standards (1967)
10. Emeneau, M.B.: India as a linguistic area. Linguistics 32, 3–16 (1956)
11. Gao, J., Suzuki, H.: Unsupervised learning of dependency structure for language modeling. In: ACL 2003, pp. 521–528. Association for Computational Linguistics (2003)
12. Klein, D.: The Unsupervised Learning of Natural Language Structure. PhD thesis, Stanford University (2004)
13. McDonald, R.: Discriminative learning and spanning tree algorithms for dependency parsing. PhD thesis, University of Pennsylvania (2006)
14. McDonald, R., Crammer, K., Pereira, F.: Online large-margin training of dependency parsers. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL) (2005)
15. McDonald, R., Satta, G.: On the complexity of non-projective data-driven dependency parsing. In: Proceedings of the International Conference on Parsing Technologies (IWPT) (2007)
16. Nivre, J.: An efficient algorithm for projective dependency parsing. In: Proceedings of International Workshop on Parsing Technologies, pp. 149–160 (2003)
17. Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., Yuret, D.: The CoNLL 2007 shared task on dependency parsing. In: Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 915–932 (2007)
18. Paskin, M.A.: Grammatical bigrams. In: Proceedings of NIPS, pp. 91–97 (2001)
19. Avinesh, P.V.S., Karthik, G.: Part-of-speech tagging and chunking using conditional random fields and transformation based learning. In: Proceedings of the IJCAI 2007 Workshop on Shallow Parsing in South Asian Languages, Hyderabad, India (2007)
20. Rao, D., Yarowsky, D.: Part of speech tagging and shallow parsing for indian languages. In: Proceedings of the IJCAI-07 Workshop on Shallow Parsing in South Asian Languages, Hyderabad, India (2007)
21. Smith, D.A., Eisner, J.: Bootstrapping feature-rich dependency parsers with entropic priors. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pp. 667–677
22. Smith, D.A., Smith, N.A.: Probabilistic models of nonprojective dependency trees. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pp. 132–140
23. Smith,N.A.: Discovery of linguistic relations using lexical attraction. PhD thesis (1998)
24. Yoshinaga, N., Miyao, Y., Torisawa, K., Tsujii, J.: Efficient LTAG parsing using HPSG parsers. In: Proc. of PACLING, pp. 342–351 (2001)
25. Yuret,D.: Discovery of linguistic relations using lexical attraction. PhD thesis (1998)