UCSG: A Wide Coverage Shallow Parsing System

G. Bharadwaja Kumar and **Kavi Narayana Murthy**
Department of Computer and Information Sciences
University of Hyderabad
g_vijayabharadwaj@yahoo.com, knmuh@yahoo.com

Abstract

In this paper, we propose an architecture, called UCSG Shallow Parsing Architecture, for building wide coverage shallow parsers by using a judicious combination of linguistic and statistical techniques without need for large amount of parsed training corpus to start with. We only need a large POS tagged corpus. A parsed corpus can be developed using the architecture with minimal manual effort, and such a corpus can be used for evaluation as also for performance improvement. The UCSG architecture is designed to be extended into a full parsing system but the current work is limited to chunking and obtaining appropriate chunk sequences for a given sentence. In the UCSG architecture, a Finite State Grammar is designed to accept *all* possible chunks, referred to as word groups here. A separate statistical component, encoded in HMMs (Hidden Markov Model), has been used to rate and rank the word groups so produced. Note that we are not pruning, we are only rating and ranking the word groups already obtained. Then we use a Best First Search strategy to produce parse outputs in best first order, without compromising on the ability to produce all possible parses in principle. We propose a bootstrapping strategy for improving HMM parameters and hence the performance of the parser as a whole.

A wide coverage shallow parser has been implemented for English starting from the British National Corpus, a nearly 100 Million word POS tagged corpus. Note that the corpus is not a parsed corpus. Also, there are tagging errors, multiple tags assigned in many cases, and some words have not been tagged. A dictionary of 138,000 words with frequency counts for each word in each tag has been built. Extensive experiments have been carried out to evaluate the performance of the various modules. We work with large data sets and performance obtained is encouraging. A manually checked parsed corpus of 4000 sentences has also been developed and used to improve the parsing performance further. The entire system has been implemented in Perl under Linux.

Key Words:- Chunking, Shallow Parsing, Finite State Grammar, HMM, Best First Search

1 Introduction

In recent times, there has been an increasing interest in wide coverage and robust but shallow parsing systems. Shallow parsing is the task of recovering only a limited amount of syntactic information from natural language sentences. Often shallow parsing is restricted to finding phrases in sentences, in which case it is also called chunking. Steve Abney (Abney, 1991) has described chunking as *finding syntactically related non-overlapping groups of words.* In CoNLL chunking task (Tjong Kim Sang and Buchholz, 2000) chunking was defined as *the task of dividing a text into syntactically non-overlapping phrases.*

Most of the shallow parsers and chunkers described in literature (Tjong Kim Sang and Buchholz, 2000; Carreras and Marquez, 2003; Dejean, 2002; Molina and Pla, 2002; Osborne, 2002; Sang, 2002; Abney, 1996; Grefenstette, 1996; Roche, 1997) have used either only rule based techniques or only machine learning techniques. Hand-crafting rules in the linguistic approach can be very laborious and time consuming. Parsers tend to produce a large number of possible parse outputs and in the absence

of suitable rating and ranking mechanisms, selecting the right parse can be very difficult. Statistical learning systems, on the other hand, require large and representative parsed corpora for training, and such training corpora are not always available. Perhaps only a good combination of linguistic and statistical approaches can give us the best results with minimal effort.

Other important observations from literature that motivated the present work are: 1) Most chunking systems have so far been tested only on small scale data 2) Good performance has been obtained only under restricted conditions 3) Performance is often evaluated in terms of individual chunks rather than complete chunk sequences for a whole sentence, and 4) Many chunkers produce only one output, not all possible outputs in some ranked order.

2 UCSG Shallow Parsing Architecture

UCSG shallow parsing architecture is set within the UCSG full parsing framework for parsing natural language sentences which was initiated in the early 1990's at University of Hyderabad by Kavi Narayana Murthy (Murthy, 1995). In this paper, the focus is only on chunking - identifying chunks or word groups, handling ambiguities, and producing parses (chunk sequences) for given sentences. This can be extended to include thematic role assignment and clause structure analysis leading towards a full parser. Figure 1 shows the basic UCSG Shallow Parsing Architecture (Kumar and Murthy, 2006).

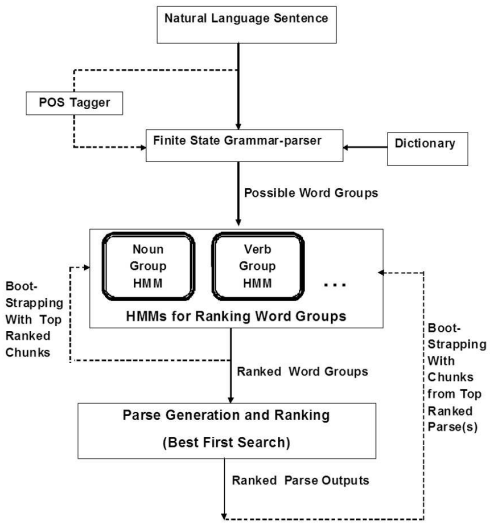

Figure 1: UCSG Shallow Parsing Architecture

The input to the parsing system is one sentence, either plain or POS tagged. Output is an ordered set of parses. Here by parse we mean a sequence of chunks that covers the given sentence with no overlaps or gaps. The aim is to produce all possible parses in ranked order hoping to get the best parse to the top.

A chunk or a "word group" as we prefer to call it in UCSG, is "a structural unit, a non-overlapping and non-recursive sequence of words, that can as a whole, play a role in some predication" (Murthy, 1995). Note that word groups do not include clauses (relative clauses, for example) or whole sentences. Every word group has a head which defines the type of the group. These word groups thus seem to be similar to *chunks* as generally understood (Molina and Pla, 2002; Sang and Buchholz, 2000; Megyesi, 2002). However, chunks in UCSG are required to correspond to thematic roles, which means for example, that prepositional phrases are handled properly. Many chunkers do not even build prepositional phrases - prepositions are treated as individual chunks in their own right. Thematic roles can be viewed from question-answering perspective. For example, in the sentence *'I teach at University of Hyderabad'*, *'at University of Hyderabad'* answers the 'where' question and should therefore be treated as a single chunk. It is well known that prepositional phrase attachment is a hard problem and the task we have set for ourselves here is thus significantly more challenging. The parse outputs in UCSG would be more semantic and hence should be better suited for many NLP applications.

In UCSG, a Finite State Grammar-Parser system generates all possible chunks in linear time. Chunk level HMMs are then used to rate and rank the chunks so produced. Finally, a kind of best first search strategy is applied to obtain chunk sequences hopefully in best first order. The aim is to develop wide coverage, robust parsing systems without need for a large scale parsed corpus to start with. Only a large POS tagged corpus is needed and a parsed corpus can be generated from within the architecture with minimal manual effort. Such a parsed corpus can be used for evaluation as also for further performance improvements.

We will need a dictionary which includes the frequency of occurrence of each word in each possible tag. Such a dictionary can be developed using a large POS tagged corpus.

2.1 Finite State Grammar-Parser

Here the task is only to *recognize* chunks and not produce a detailed description of the internal structure of chunks. Also, chunks by definition are non-recursive in nature, only linear order, repetition and optional items need to be considered. Finite state grammars efficiently capture linear precedence, repetition and optional occurrence of words in word groups. Finite state machines are thus both necessary and sufficient for recognizing word groups (Murthy, 1995). It is also well known that Finite State Machines are computationally efficient - linear time algorithms exist for recognizing word groups. All possible word groups can be obtained in a single left-to-right scan of the given sentence in linear time (Murthy, 1995). Finite state grammars are also conceptually simple and easy to develop and test.

The Finite State module accepts a sentence (either already POS tagged or tagged with all possible categories using the dictionary) and produces an unordered set of possible chunks taking into account all lexical ambiguities.

2.2 HMMs for Rating and Ranking Chunks

The second module is a set of Hidden Markov Models (HMMs) used for rating and ranking the word groups already produced by the Finite State Grammar-Parser. The hope is to get the best chunks near the top. This way, we are not pruning and yet we can hope to get the right chunks near the top and push down the others.

Words are observation symbols and POS tags are states in our HMMs. Formally, a HMM model $\lambda = (\pi, A, B)$ for a given chunk type can be described as follows:

Number of States (N) = number of relevant POS Categories

Number of Observation Symbols (M) = number of Words of relevant categories in the language

The initial state probability

$$\pi_i = P\{q_1 = i\} \tag{1}$$

where $1 \leq i \leq N$, q_1 is a category (state) starting a particular word group type.

State transition probability

$$a_{ij} = P\{q_{t+1} = j | q_t = i\} \tag{2}$$

where $1 \leq i, j \leq N$ and q_t denotes the category at time t and q_{t+1} denotes the category at time t+1.

Observation or emission probability

$$b_j(k) = P\{o_t = v_k | q_t = j\} \tag{3}$$

where $1 \leq j \leq N$, $1 \leq k \leq M$ and v_k denotes the k^{th} word, and q_t the current state.

We first pass a large POS tagged corpus through the Finite State module and obtain all possible chunks. Taking these chunks to be equi-probable, we estimate the HMM parameters by taking the ratios of frequency counts. One HMM is developed for each major category of chunks, say, one for noun-groups, one for verb-groups, and so on. The B matrix values are estimated from a dictionary that includes frequency counts for each word in every possible category. These initial models of HMMs are later refined using a bootstrapping technique as described later.

We simply estimate the probability of each chunk using the following equation :

$$P(O, Q|\lambda) = \pi_{q_1} b_{q_1}(o_1) a_{q_1,q_2} b_{q_2}(o_2) a_{q_2,q_3} \cdots a_{q_{t-1},q_t} b_{q_t}(o_t)$$

where q_1 ,q_2, $\cdots$, q_t is a state sequence, o_1 , o_2,$\cdots$, o_t is an observation sequence. Note that no Viterbi search involved here and the state sequence is also known. Thus even Forward/Backward algorithm is not required and rating the chunks is therefore computationally efficient.

The aim here is to assign the highest rank for the correct chunk and to push down other chunks. Since a final parse is a sequence of chunks that covers the given sentence with no overlaps or gaps, we evaluate the alternatives at each position in the sentence in a left-to-right manner.

Here, we use *Mean Rank Score* to evaluate the performance of the HMMs. Mean Rank Score is the mean of the distribution of ranks of correct chunks produced for a given training corpus. Ideally, all correct chunks would be at the top and hence the score would be 1. The aim is to get a Mean Rank Score as close to 1 as possible.

2.3 Parse Generation and Ranking

Parsing is a computationally complex task and generating all possible parses may be practically difficult. That is why, a generate-and-test approach

where we first generate all possible parses and then look for the correct parse among the parses produced is impracticable. Simply producing all or some parses in some random or arbitrary order is also not of much practical use. Many chunkers produce a single output which may or may not be correct. Here we instead propose a best first strategy wherein the very production of possible parses is in best first order and so, hopefully, we will get the correct parse within the top few and in practice we need not actually generate all possible parses at all. This way, we overcome the problems of computational complexity and at the same time avoid the risk of missing the correct parse if pruning is resorted to. Performance can be measured not only in terms of percentage of input sentences for which a fully correct parse is produced but also in terms of the rank of the correct parse in the top k parses produced, for any chosen value of k.

It may be noted that although we have already rated and ranked the chunks, simply choosing the locally best chunks at each position in a given sentence does not necessarily give us the best parse (chunk sequence) in all cases. Hence, we have mapped our parse selection problem into a graph search problem and used best first search algorithm to get the best parse for a given sentence.

Words and chunks in a sentence are referred to in terms of the positions they occupy in the sentence. Positions are marked between words, starting from zero to the left of the first word. The positions in the sentence are treated as nodes of the resulting graph. If a sentence contains N words then the graph contains $N + 1$ nodes corresponding to the $N + 1$ positions in the sentence. Word group $W_{i,j}$ is represented as an edge form node i to node j. We thus have a lattice structure. The cost of a given edge is estimated from the probabilities given by the HMMs. If and where a parsed training corpus is available, we can also use the transition probability from previous word group type to current word group type. It is possible to use the system itself to parse sentences and from that produce a manually checked parsed corpus with minimal human effort. We always start from the initial node 0. N is the goal node. Now our parse selection problem for a sentence containing N words becomes the task of finding an optimal (lowest cost) path from node 0 to node N.

We use the standard best first search algorithm. In best first search, we can inspect all the currently-available nodes, rank them on the basis of our partial knowledge and select the most promising of the nodes. We then expand the chosen node to generate it successors. The worst case complexity of best first search algorithm is exponential: $O(b^m)$, where b is the branching factor (i.e., the average number of nodes added to the open list at each level), and m is the maximum length of any path in the search space. As an example, a 40 word sentence has been shown to produce more than 10^{15} different parses (Kumar, 2007). In practice, however, we are usually interested in only the top k parses for some k and exhaustive search is not called for.

2.4 Bootstrapping

The HMM parameters can be refined through bootstrapping. We work with large data sets running into many hundreds of thousands of sentences and Baum-Welch parameter re-estimation would not be very practical. Instead, we use parsed outputs to re-build HMMs. By parsing a given sentence using the system and taking the top few parses only as training data, we can re-build HMMs that will hopefully be better. We can also simply use the top-ranked chunks for re-building the HMMs. This would reduce the proportion of invalid chunks in the training data and hence hopefully result in better HMM parameters. As can be seen from the results in the next section, this idea actually works and we can significantly improve the HMM parameters and improve parser performance as well.

3 Experiments and Results

The entire parsing system has been implemented in Perl under Linux. Extensive experimentation has been carried out to evaluate the performance of the system. However, direct comparisons with other chunkers and parsers are not feasible as the architectures are quite different. All the experiments have been carried out on a system with Pentium Core 2 DUO 1.86 GHz Processor and 1 GB RAM. Transcripts from the implemented system have been included in the next section.

3.1 Dictionary

We have developed a dictionary of 138,000 words including frequency of occurrence for each tag for each word. The dictionary includes derived words but not inflected forms. The dictionary has been built from the British National Corpus(BNC) (Burnard, 2000), an English text corpus of about 100 Million words. Closed class words have been manually checked. The dictionary has a coverage of 98% on the BNC corpus itself, 86% on the Reuters News Corpus (Rose et

al., 2002) (about 180 Million words in size), 96.36% on the Susanne parsed corpus (Sampson, 1995) and 95.27% on the Link parser dictionary.

3.2 Sentence Boundary Detection

We have developed a sentence segmentation module using the BNC corpus as training data. We have used *delimiter, prefix, suffix and after-word* as features and extracted patterns from the BNC corpus. Decision Tree algorithms have been used and an average F-Measure of 98.70% has been obtained, comparable to other published results. See (Htay et al., 2006) for more details.

3.3 Tag Set

We have studied various tag sets including BNC C5, BNC C7, Susanne and Penn Tree Bank tag sets. Since our work is based on BNC 1996 edition with C5 tag set, we have used C5 tag set and made some extensions as required. We now have a total of 71 tags in our extended tag set (Kumar, 2007).

3.4 Manually Parsed Corpus

We have developed a manually checked parsed corpus of 4000 sentences, covering a wide variety of sentence structures. Of these, 1000 sentences have been randomly selected from the BNC corpus, 1065 sentences from 'Guide to Patterns and Usage in English' (Hornby, 1975) and 1935 sentences from the CoNLL-2000 test data. This corpus is thus very useful for evaluating the various modules of the parsing architecture and also for bootstrapping.

This corpus was developed by parsing the sentences using this UCSG shallow parser itself and then manually checking the top parse and making corrections where required. Our experience shows that this way we can build manually checked parsed corpora with minimal human effort.

3.5 Tagging

If a POS tagger is available, we can POS tag the input sentences before sending them to the parser. Otherwise, all possible tags from the dictionary may be considered. In our work here, we have not used any POS tagger. All possible tags are assigned from our dictionary and a few major rules of inflectional morphology of English, including plurals for nouns, past tense, gerundial and participial forms of verbs and degrees of comparison for adjectives are handled. Unresolved words are assigned NP0 (Proper Name) tag.

3.6 Finite State Grammar

We have developed a Finite State Grammar for identifying English word groups. The Finite State Machine has a total of 50 states of which 24 are final states. See (Kumar, 2007) for further details.

The UCSG Finite State Grammar recognizes verb-groups, noun-groups, adverbial-groups, adjective-groups, to-infinitives, coordinate and subordinate conjunctions. There are no separate prepositional phrases - prepositions are treated as surface case markers in UCSG - their primary role is to indicate the relationships between chunks and the thematic roles taken up by various noun groups. Prepositional groups are therefore treated on par with noun groups.

We have evaluated the performance of the FSM module on various corpora - Susanne Parsed Corpus, CoNLL 2000 test data set and on our manually parsed corpus of 4000 sentences. The evaluation criteria is Recall (the percentage of correct chunks recognized) alone since the aim here is only to include the correct chunks. We have achieved a high recall of 99.5% on manually parsed corpus, 95.06% on CoNLL test data and 88.02% on Susanne corpus.

The reason for the relatively low Recall on the Susanne corpus is because of the variations in the definition of phrases in Susanne corpus. For example, Susanne corpus includes relative clauses into noun groups. The reasons for failures on CoNLL test data have been traced mainly to missing dictionary entries and inability of the current system to handle multi-token adverbs.

3.7 Building and Refining HMMs

HMMs were initially developed from 3.7 Million POS-tagged sentences taken from the BNC corpus. Sentences with more than 40 words were excluded. Since we use an extended C5 tag set, POS tags had to be mapped to the extended set where necessary. HMM parameters were estimated from the chunks produced by the Finite State grammar, taking all chunks to be equi-probable. Separate HMMs were built for noun groups, verb groups, adjective groups, adverb groups, infinitive groups and one HMM for all other chunk types.

The chunks produced by the FSM are ranked using these HMMs. It is interesting to observe the Recall and Mean Rank Score within the top k ranks, where k is a given cutoff rank. Table 1 shows that there is a clear tendency for the correct chunks to bubble up

close to the top. For example, more than 95% of the correct chunks were found within the top 5 ranks.

Table 1: Performance of the HMM Module on the Manually Parsed Corpus of 4000 sentences

Cut-off	Plain		POS Tagged	
	Mean Rank	Cumulative Recall (%)	Mean Rank	Cumulative Recall (%)
1	1	43.06	1	62.74
2	1.38	69.50	1.28	86.97
3	1.67	84.72	1.43	95.64
4	1.85	91.69	1.50	98.31
5	1.96	95.13	1.54	99.25

We have also carried out some experiments to see the effect of the size of training data used to build HMMs. We have found that as we use more and more training data, the HMM performance improves significantly, clearly showing the need for working with very large data sets. See (Kumar, 2007) for more details.

3.7.1 Bootstrapping

To prove the bootstrapping hypothesis, we have carried out several experiments. Plain text sentences from BNC corpus, 5 to 20 words in length, have been used. All possible chunks are obtained using the Finite State Grammar-Parser and HMMs built from these chunks. In one experiment, only the chunks rated highest by these very HMMs are taken as training data for bootstrapping. In a second experiment, best first search is also carried out and chunks from the top ranked parse alone are taken for bootstrapping. In a third experiment, data from these two sources have been combined. Best results were obtained when the chunks from the top parse alone were used for bootstrapping. Table 2 shows the effect of bootstrapping on the HMM module for plain sentences.

Table 2: Effect of Bootstrapping: on 4000 sentences from Manually Parsed Corpus containing a total of 27703 chunks

Cutoff	Iteration-1		Iteration-2	
	Recall	Mean Rank	Recall	Mean Rank
1	45.52	1.0	47.25	1.0
2	71.43	1.36	72.81	1.35
3	85.22	1.63	85.95	1.60
4	91.75	1.80	92.20	1.77
5	94.94	1.90	95.30	1.87

It may be observed that both the Recall and Mean Rank Scores have improved. Our experiments show that there is also some improvement in the final parse when the HMMs obtained through bootstrapping are used. These observations, seen consistently for both plain and POS tagged sentences, show the effectiveness of the overall idea.

3.8 Parse Generation and Ranking

It may be noted that in principle the performance of the parser in terms of its ability to produce the correct parse is limited only by the Finite State Grammar and the dictionary, since the other modules in the UCSG architecture do not resort to any pruning. However, in practical usage we generally impose a time limit or a cutoff and attempt to produce only the top k parses. In this latter case, the percentage of cases where the fully correct parse is included would be a relevant performance indicator. Percentage of correct chunks in the top parse is another useful indicator.

When tested on untagged sentences, on the 1065 linguistically rich sentence corpus forming part of the manually checked parsed corpus developed by us, the parser could generate fully correct parse within the top 5 parses in 930 cases, that is, 87.32% of the cases. In 683 cases the correct parse was the top parse, 146 correct parses were found in position 2, 56 in position 3, 29 in position 4 and 16 in position 5. Thus the mean rank of the correct parses is 1.44. There is a clear tendency for the correct parses to appear close to the top, thereby verifying the best first strategy. If top 10 parses are generated, correct parse is obtained in 52 more cases and the Mean Rank Score goes to 1.75.

We give below the performance on the whole of our 4000 strong manually checked corpus. Plain sentences and POS tagged sentences have been tested separately. The results are summarized in table 3. Here, we have restricted the parsing time taken by the best first search algorithm to 3 epoch seconds for each sentence.

Table 3: Performance of the Best First Search Module - Test Data of 4000 Sentences

Rank	No. of Correct Parses	
	(Plain)	(POS tagged)
1	1210	2193
2	352	495
3	157	164
4	83	129
5	68	91
% of Correct Parses in Top 5	46.75	76.80
% of Correct Chunks in Best Parse	83.92	88.26

In about 77% of the cases, the fully correct parse is found within the top 5 parses when the input sentences are POS tagged. Given the nature of chunks produced in UCSG, this is quite encouraging. In fact the top parse is nearly correct in many cases. Further experiments and manual evaluations are planned.

We have also observed that 96.01% of the words are assigned correct POS tags in the top parse. We observe that most of the times the top parse given by the parse generation module is almost correct.

Chunkers are usually evaluated just for the percentage of correct chunks they produce. We have placed greater demands on ourselves and we expect our parser to produce optimal chunk sequence for the whole sentence. Further, we produce all (or top few) combinations and that too in hopefully a best first order. Also, the very nature of chunks in UCSG makes the task more challenging. More over, we have used a fairly fine grained tag set with more than 70 tags. The data we have started with, namely the BNC POS tagged corpus, is far from perfect. Given these factors, the performance we are able to achieve both in terms of percentage of correct chunks in the top parse and rank of the fully correct parse is very encouraging.

4 Transcripts:

Here we give the actual transcripts from the system. For want of space, only a very simple example has been included. Stars have been added in the beginning of lines containing correct alternatives.

```
Input: I am studying at University of Hyderabad.

Tags from the Dictionary: <PNN_CRD><i>##<VBB><am>##<VVG>
    <studying>##<PRN_PRP_AVP><at>##<NN1><university>
    ##<PRN_PRF_AVP><of>##<NP0><Hyderabad>##

Chunks Recognized by the FSM:
<ng><0-1><CRD><i>
<ajg><0-1><CRD><i>
*<ng><0-1><PNN><i>
<vg><1-2><VBB><am>
*<vg><1-3><VBB><am>##<VVG><studying>
<vg><1-4><VBB><am>##<VVG><studying>##<AVP><at>
<vgs><2-3><VVG><studying>
<ng><2-3><VVG><studying>
<ajg><2-3><VVG><studying>
<vgs><2-4><VVG><studying>##<AVP><at>
<ng><2-5><VVG><studying>##<PRP><at>##<NN1><university>
<ng><2-7><VVG><studying>##<PRP><at>##<NN1><university>
    ##<PRF><of>##<NP0><hyderabad>
<part><3-4><AVP><at>
<ng><3-5><PRP><at>##<NN1><university>
*<ng><3-7><PRP><at>##<NN1><university>##<PRF><of>##
    <NP0><hyderabad>
<ng><4-5><NN1><university>
<ng><4-7><NN1><university>##<PRF><of>##<NP0><hyderabad>
<part><5-6><AVP><of>
<ng><5-7><PRF><of>##<NP0><hyderabad>
<ng><6-7><NP0><hyderabad>

Ranking by HMMs:
*<ng><0-1><PNN><i><-3.2491231040407><1><3><1>
<ng><0-1><CRD><i><-9.56376400947296><2><3><1>
<ajg><0-1><CRD><i><-36.8109739544272><3><3><1>
<vg><1-2><VBB><am><-7.27367328109116><1><3><2>
*<vg><1-3><VBB><am>##<VVG><studying><-15.945895214915>
    <2><3><2>
<vg><1-4><VBB><am>##<VVG><studying>##<AVP><at>
    <-25.5608664628101><3><3><2>
<vgs><2-3><VVG><studying><-10.5328994260119><1><6><3>
<ng><2-3><VVG><studying><-12.7929752284183><2><6><3>
<vgs><2-4><VVG><studying>##<AVP><at><-20.147870673907>
    <3><6><3>
<ng><2-5><VVG><studying>##<PRP><at>##<NN1><university>
    <-30.3473074722636><4><6><3>
<ajg><2-3><VVG><studying><-32.767076078699><5><6><3>
<ng><2-7><VVG><studying>##<PRP><at>##<NN1><university>##
    <PRF><of>##<NP0><hyderabad><-35.1643970692879><6><6><3>
<part><3-4><AVP><at><-7.99897865005313><1><3><4>
<ng><3-5><PRP><at>##<NN1><university><-15.7772256956695>
    <2><3><4>
*<ng><3-7><PRP><at>##<NN1><university>##<PRF><of>##<NP0>
    <hyderabad><-20.5943152926938><3><3><4>
<ng><4-5><NN1><university><-13.2259579687766><1><2><5>
<ng><4-7><NN1><university>##<PRF><of>##<NP0><hyderabad>
    <-18.0430475658009><2><2><5>
<part><5-6><AVP><of><-3.87313237166961><1><2><6>
<ng><5-7><PRF><of>##<NP0><hyderabad><-19.0843146188301>
    <2><2><6>
<ng><6-7><NP0><hyderabad><-3.43828759462479><1><1><7>

Final Parse:

*<ng>[<PNN><i>]</ng> <vg>[<VBB><am>##<VVG><studying>]</vg>
    <ng>[<PRP><at>##<NN1><university>##<PRF><of>##<NP0>
    <hyderabad>]</ng> -- -41.2629507152745

<ng>[<PNN><i>]</ng> <vg>[<VBB><am>]</vg> <ng>[<VVG>
    <studying>]</ng><ng>[<PRP><at>##<NN1><university>##<PRF>
    <of>##<NP0><hyderabad>]</ng> -- -46.7375549370651

<ng>[<PNN><i>]</ng> <vg>[<VBB><am>]</vg> <ng>[<VVG>
    <studying>##<PRP><at>##<NN1><university>##<PRF><of>##
    <NP0><hyderabad>]</ng> -- -47.1608105580448

<ng>[<CRD><i>]</ng> <vg>[<VBB><am>##<VVG><studying>]</vg>
    <ng>[<PRP><at>##<NN1><university>##<PRF><of>##<NP0>
    <hyderabad>]</ng> -- -47.5775916207068

<ng>[<PNN><i>]</ng> <vg>[<VBB><am>##<VVG><studying>##
    <AVP><at>]</vg><ng>[<NN1><university>##<PRF><of>##
    <NP0><hyderabad>]</ng> -- -48.3266542362767
```

5 Conclusions:

A hybrid architecture for developing wide coverage shallow parsing systems, without need for a large scale parsed corpus to start with, has been proposed and its effectiveness demonstrated by developing a wide coverage shallow parser for English. The system has been built and tested on very large data sets, covering a wide variety of texts, giving us confidence that the system will perform well on new, unseen texts. The system is general and not domain specific, but we can adapt and fine tune for any specific domain to achieve better performance. We are confident that wide coverage and robust shallow parsing systems can be developed using the UCSG architecture for other languages of the world as well. We plan to continue our work on English parsing while we also start our work on Telugu.

References

Steven P. Abney. 1991. *Parsing by Chunks*. Kluwer, Principle-Based Parsing: Computation and Psycholinguistics edition.

Steven P. Abney. 1996. Partial Parsing via Finite-State Cascades. In *Workshop on Robust Parsing, 8th European Summer School in Logic, Language and Information*, pages 8–15, Prag.

L. Burnard. 2000. The Users' Reference Guide for the British National Corpus. Oxford University Computing Services, Oxford.

Xavier Carreras and Lluys Marquez. 2003. Phrase Recognition by Filtering and Ranking with Perceptrons. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing, RANLP-2003*, pages 127–132, Borovets, Bulgaria.

Herve Dejean. 2002. Learning Rules and their Exceptions. In *Journal of Machine Learning Research, Volume 2*, pages 669–693.

G. Grefenstette. 1996. Light Parsing as Finite State Filtering. In *Workshop on Extended Finite State Models of Language*, Budapest, Hungary.

A. S. Hornby. 1975. *Guide to Patterns and Usage in English*. Oxford University Press.

Hla Hla Htay, G. Bharadwaja Kumar, and Kavi Narayana Murthy. 2006. Constructing English-Myanmar Parallel Corpora. In *Proceedings of Fourth International Conference on Computer Applications*, pages 231–238, Yangon, Myanmar.

G Bharadwaja Kumar and Kavi Narayana Murthy. 2006. UCSG Shallow Parser. *Proceedings of CICLING 2006, LNCS*, 3878:156–167.

G. Bharadwaja Kumar. 2007. *UCSG Shallow Parser: A Hybrid Architecture for a Wide Coverage Natural Language Parsing System*. Phd thesis, University of Hyderabad.

B Megyesi. 2002. Shallow Parsing with PoS Taggers and Linguistic Features. In *Journal of Machine Learning Research, Volume 2*, pages 639–668.

Antonio Molina and Ferran Pla. 2002. Shallow Parsing using Specialized HMMs. In *Journal of Machine Learning Research, Volume 2*, pages 595–613.

Kavi Narayana Murthy. 1995. *Universal Clause Structure Grammar*. Phd thesis, University of Hyderabad.

Miles Osborne. 2002. Shallow Parsing using Noisy and Non-Stationary Training Material. In *Journal of Machine Learning Research, Volume 2*, pages 695–719.

E. Roche. 1997. *Parsing with Finite State Transducers*. MIT Press, finite-State Language Processing edition.

T.G. Rose, M. Stevenson, and M. Whitehead. 2002. The Reuters Corpus Volume 1 - from Yesterday's News to Tomorrow's Language Resources. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, Las Palmas de Gran Canaria.

Geoffrey Sampson. 1995. *English for the Computer*. Clarendon Press (The Scholarly Imprint of Oxford University Press).

E. F. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132, Lisbon, Portugal.

Erik F. Tjong Kim Sang. 2002. Memory-Based Shallow Parsing. In *Journal of Machine Learning Research, Volume 2*, pages 559–594.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 Shared Task: Chunking. In Claire Cardie, Walter Daelemans, Claire Nedellec, and Erik Tjong Kim Sang, editors, *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132. Lisbon, Portugal.