Development of Tree-bank Based Probabilistic Grammar for Urdu Language

[1]Qaiser Abbas, [2]Nayyara Karamat, [3]Sadia Niazi
[1]Department of Computer Science NUCES/FAST, Lahore, PK , [2]CRULP Center, NUCES/FAST, Lahore. PK,
[3]Department of Psychology, UOS, Sargodha,PK
[1]qaiser.abbas@uos.edu.pk, [2]nayyara.karamat@nu.edu.pk, [3]sadia_niazi2003@yahoo.com

***Abstract*-- The process includes in hand tagged corpus, tree annotation on paper for large corpus, NU-FAST Treebank in form of brackets, extraction of CFG through NU-FAST Treebank, evaluation of PCFG from CFG and then PDCG from PCFG for inspection/testing through PROLOG parser.**

I. INTRODUCTION

It will be injustice to not talk about Penn Treebank, when we are discussing treebank based PCFG's. This project was originated in 1989 and it contained 4.5 million of words. During first three years, only POS tagging was done automatically and manually, and speed, consistency and accuracy were achieved through this process (Beatrice Santorini, 1990). The POS tagged sentences were partially parsed and skeletal syntactic representation was made which was corrected manually and then bracketing process was performed (Mitchell P. Marcus et al. 1993). The resulting Penn Treebank, is widely used in natural language processing, speech recognition, and integrated spoken language systems as well as in theoretical linguistics. Fidditch parser (Hindle 1983, Hindle 1989) was used for initial parsing of the sentences and then annotators used a mouse based GUI to correct the parse manually. The Penn Treebank syntactic tagset is given in the table III on page 10(Mitchell P. Marcus et. al. 1993), however a detailed syntactic material can be seen in (Santorini and Marcinkiewicz, 1991). The tags and null elements given in the table III mentioned above are elaborated with their respective symbols; however, the null elements are the extra features which can exploit the sentence's predicate argument structure (Mitchell Marcus, Grace Kim, et al. 1994) and can determine verb transitivity. The Penn Treebank consists of 4,885,798 POS tagged words and 2,881, 188 words with skeletal parsing. (Mitchell P. Marcus et al. 1993)

A detailed document with overview of base clause structure, notation, punctuation, null elements, pseudo attach, copular verbs, coordination, shared complements and modifiers for in coordinate structure, wh-phrases, subordinate clauses, modification of NP, titles, gerund and participles, infinitives, small clauses and their near relatives, cleft, it extra position and much more are included in revised complete document in (Ann Bies, Mark Feguson, et al. 1995). In this document features mentioned above are presented with examples of bracketed text.

The Penn Treebank is used in many researches. An important one is work on probablistic context free grammars, PCFG (Eugene Charniak, 1996). A CFG is extracted from the Penn Treebank and then a PCFG is obtained to parse sentences. The formula used to calculate PCFG is given below.

$$P(r) = \frac{|r|}{\sum_{r' \in \{r' | \lambda(r') = \lambda(r)\}} |r'|} \qquad (1)$$

Eugene also introduces some new tags like ORD, PRT, aux and auxg etc. in Treebank during CFG extraction process. A total of 10,605 rules are extracted, among them 3942 were repeated but none of the rules are ignored. HMM Viterbi algorithm is used to find the most probable parse (map parse). Precision, Recall and Accuracy measures are used to measure the results, it has been observed that precision, recall and accuracy measures were high for short sentences and decrease with the length of sentences. However, 88.6, 91.7 and 97.9 percent precision, recall and accuracy respectively were achieved by comparing the map parse with Treebank test data(Eugene Charniak,1996).

Another test was also made on the repeated subset rules and found that the parsing results are almost the same. Moreover, the results are one percent more accurate with full size CFG rules rather than reduced size CFG rules.

A PCFG has all the qualities of CFG with an additional conditional probability for each rule such that P(A→B|A), where A is left hand side terminal and B is right hand side expansion (Daniel Jurafsky and James H. Martin. 1999). It is also important that the sum of expansion of a particular non terminal should be 1. The probability of the tree for a sentence can be calculated simply by multiplying all the derivations of the tree. It is also possible that more than one parses are produced with their respective probabilities from the given PCFG. Here the algorithm used for parsing plays an important role and picks the best one among them by using the most likely method.

Definite Clause Grammar was introduced by (Pereira and Warren1980) which is used to express CFG in PROLOG. However, the traditions are updated now and we can even handle PCFG using DCG in PROLOG for example the PCFG rule NP→D N /0.8 is expressed in DCG representation as np(X) → d(Y), n(Z), {X is 0.8*Y*Z}. This DCG representation with probabilities is also known as PDCG (Tony Abou-Assaleh, Nick Cercone, Vlado Keselj, 2003). Queries can be passed to view the parse trees with probabilities as: ?- s([time, flies, like, an, arrow], [], T). and then trees will be displayed like mentioned below:

(0.0084) T = s(np(n(time)), vp(v(flies), pp(p(like), np(d(an), n(arrow)))))
(0.00036) T = s(np(n(time), n(flies)), vp(v(like), np(d(an), n(arrow))))

II. Design

An already POS tagged corpus (Hassan Sajid, 2007) is used to build Urdu TreeBank. However, some of its tags are changed to obtain well defined grammar rules. For example, a single tag Particle(P) was used for words کا، کی، کے، کو، نے، میں، تلک، پر etc. Since the tagger treats all these words as particle, further classification was needed to define finer rules at the phrase level. These particles are subdivided into case markers, preposition, genitive marker and Kaf pronoun. It is also pertinent to note that sometimes these particles behavior at semantic level determines their correct tag which makes the decision difficult. Some of the tags from syntactic tagset are given in table I. The syntactic tagset has the tags which have "S" level attachment and can also be attached at higher level in a tree.

Grammar analysis for the Urdu TreeBank is largely taken from a grammar development work done at CRULP (Urdu Grammar Rules, 2005).

TABLE I
SYNTACTIC TAGSET FOR NU-FAST TREE-BANK

Tags	Description
KP	Kaf Pronoun Phrase
NP	Noun Phrase
VP	Verb Phrase
GP	Genitive Phrase
WALAP	WALA Phrase to handle word والا، والی etc. effects.
VPINF	Verb Phrase Infinitive
ADJP	Adjective Phrase
NEGP	Negative Phrase
QP	Quantifier Phrase due to its length in Urdu Language
SC	Sentence Coordination use at sentence level attachment.
CAP	Cardinal Phrase
UP	Unit Phrase
VBP	Special Case of Verb Phrase
ADVP	Adverb Phrase
AAUXP	Aspect Auxiliary Phrase
TAUXP	Tense Auxiliary Phrase
SEP	Se Phrase to handle word سے (se) effects
PREP	Prepositional Phrase
KERP	KER Phrase to handle word کر effects
S	Start of sentence or sub sentence
DATEP	Date Phrase
NN	Special case of noun like وغیرہ (waghaira)
SM	Sentence Marker
QWP	Question Word Phrase
AKP	Special case of Adverbial Kaf Pronoun
ADV	Special case of Adverbs
KPR	KPR is used to differentiate it from KP tagged word like (kaun) کون

Following guidelines are followed during tree annotation of tagged sentences.

1) Word reordering is an important clue to tree hierarchy.

2) It was observed at first, that GP(genitive phrase) is unbreakable but later it is concluded that GP depends on agreement, if the agreement occurs then we can not break GP, otherwise GP can be broken like sentence no. 81 in NU-FAST-Treebank below.

3) The phrase structure dependencies and in some cases word dependencies can be concluded from semantics.

4) Intensifiers are kept flat with the running phrase.

5) Some special nouns like وغیرہ (etc.) are kept flat at the sentence level and all the NN's except special cases are converted into NP first and then move forward.

6) When a NN becomes part of the verb phrase, then it eradicated the qualities of taking argument e.g. بےجا استعمال کرتی ہے۔ (extra ordinary uses).

7) تک،تلک،سے،پر،میں like words are declared as Prepositions.

8) میں (in, into), sometimes behaves like preposition like افریقہ کے جنگل میں بندوق سے (with a gun in the jungle of Africa) and sometime behaves like case marker or particle like محکمے میں شامل (included in the department), so decision of preposition phrase and other phrase is made semantically.

9) The independence of a phrase is concluded by ignoring it from the sentence.

10) Verb phrases are categorized into VPINF and TAUXP and AAUXP

11) The words with letter نے (Ne) like پکڑنے (pakarney) requires intention because it mostly depends on the following and preceding words like sentence no. 25 in the NU-FAST Treebank as below.

12) The other regional languages also help out in some critical situations like انہی میں سے (from those) can be interpreted in Punjabi as انہاں وچوں (from those) which gives a clue that last two words from the these three are a unit.

13) Phrase boundary identification test is useful.

14) When a NN noun has no agreement with the VB in the verb phrase then we can join the NN noun with the VB and hence producing a VPINF.

15) Same is rule for Adjective ADJ as in above

So a huge set of rules are applied during manual annotation of sentences into tree forms, which is then recorded into bracket forms manually with a complete inspection and verification. So the output of this all exercise is NU-FAST Treebank with 14214 words and 1011 sentences. This Treebank includes data from a tagged news corpus available at CRULP (www.crulp.org). A sample bracketed sentence from the NU-FAST Treebank is given in Fig. 1 below.

```
اس<PD>لئے<NN>80<CA>فیصد<ADV>بچوں<NN>کو<P>تربیت<NN>اور<CC>
دوڑ<NN>کے<P>دوران<NN>شدید<ADJ>چوٹیں<NN>بھی<I>آتی<VB>ہیں<TA>-<S
M>
( S ( NP ( PD اس ) ( NN لئے ) )
    ( KP ( NP ( ADVP ( CA 80 ) ( ADV فیصد ) ) ( NN بچوں ) )
          ( P کو ) )
    ( NP ( GP ( NP ( NN تربیت ) ( CC اور ) ( NN دوڑ ) ) ( P
          کے ) ) ( NN درمیان ) )
     ( NP ( ADJ شدید ) ( NN چوٹیں ) ( I بھی ) )
      ( VP ( VB آتی ) ( TA ہیں ) )
      ( SM . )
)
```

Fig. 1. A bracketed sentence from NU-FAST Treebank

The work after this stage is done through an automated process. A CFG is extracted from the bracketed Treebank with the help of stack based approach. For example bracket form of NP → ADJ NN is (NP(ADJ)(NN)). This process gives a CFG with 22376 grammar rules, more than double the rules proposed by (Eugene Charniak,1996). The rules more than once occurred in the extracted grammar, but since it gives improvement as proposed by (Eugene Charniak,1996). So, it is not removed from the overall grammar. It is easy to conclude that since the lexicon is involved in CFG, so the number ratio of repeated rules is high and it is almost equal to the 1/4th of total number of rules. The sample CFG of the above sample sentence is given Fig. 2 below.

PD --> اس	KP --> NP P	NN --> چوٹیں
NN --> لیے	NN --> تربیت	I --> بھی
NP --> PD NN	CC --> اور	NP --> ADJ NN I
CA --> 80	NN --> دوڑ	VB --> آتی
ADV --> فیصد	NP --> NN CC NN	TA --> ہیں
ADVP --> CA ADV	P --> کے	VP --> VB TA
NN --> بچوں	GP --> NP P	SM --> .
NP --> ADVP NN	NN --> درمیان	S --> NP KP NP NP
P --> کو	NP --> GP NN	VP SM
	ADJ --> شدید	

Fig. 2. CFG extracted from NU-FAST Treebank

The next phase of this research project was probability calculation of CFG rules which resulted in the same set of grammar rules with probabilities. Probabilities are calculated by the same procedure as applied in (Eugene Charniak, 1996). Since the actual probabilities of below mentioned sentence in NU-FAST Treebank is very much low, however, to understand the concept, we run the probability calculation automated process specific to this sentence which gives a PCFG of this sentence as mentioned in the fig. 3 below.

PD --> اس	[1.000000]
NN --> لیے	[0.166667]
NP --> PD NN	[0.200000]
CA --> 80	[1.000000]
ADV --> فیصد	[1.000000]
ADVP --> CA ADV	[1.000000]
NN --> بچوں	[0.166667]
NP --> ADVP NN	[0.200000]
P --> کو	[0.500000]
KP --> NP P	[1.000000]
NN --> تربیت	[0.166667]
CC --> اور	[1.000000]
NN --> دوڑ	[0.166667]
NP --> NN CC NN	[0.200000]
P --> کے	[0.500000]
GP --> NP P	[1.000000]
NN --> درمیان	[0.166667]
NP --> GP NN	[0.200000]
ADJ --> شدید	[1.000000]
NN --> چوٹیں	[0.166667]
I --> بھی	[1.000000]
NP --> ADJ NN I	[0.200000]
VB --> آتی	[1.000000]
TA --> ہیں	[1.000000]
VP --> VB TA	[1.000000]
SM --> .	[1.000000]
S --> NP KP NP NP VP SM	[1.000000]

Fig. 3. A sample of PCFG calculated from an automated process of NU-FAST Treebank

It is beyond the scope of the project at that time to build a parser; however, parsing of sentences according to PCFG developed is necessary to gauge the whole work done. PROLOG can be used for this purpose, which can parse the sentences with probability and without probability if a DCG (Definite Clause Grammar) is provided. The pattern of DCG is studied in PROLOG and an automated approach is adopted which converts our PCFG data into PDCG format recommended for PROLOG. A sample of PDCG format of the sentence is given in Fig. 4 below.

```
s(P0, s(NP, KP, NP, NP, VP, SM)) --> np(P1, NP), kp(P2, KP), np(P3, NP), np(P4,
NP), vp(P5, VP), sm(P6, SM), {P0 is P1*P2*P3*P4*P5*P6*1.000000}.
advp(P0, advp(CA, ADV)) --> ca(P1, CA), adv(P2, ADV), {P0 is P1*P2*1.000000}.
gp(P0, gp(NP, P)) --> np(P1, NP), p(P2, P), {P0 is P1*P2*1.000000}.
kp(P0, kp(NP, P)) --> np(P1, NP), p(P2, P), {P0 is P1*P2*1.000000}.
np(P0, np(ADJ, NN, I)) --> adj(P1, ADJ), nn(P2, NN), i(P3, I), {P0 is
P1*P2*P3*0.200000}.
np(P0, np(ADVP, NN)) --> advp(P1, ADVP), nn(P2, NN), {P0 is P1*P2*0.200000}.
np(P0, np(GP, NN)) --> gp(P1, GP), nn(P2, NN), {P0 is P1*P2*0.200000}.
np(P0, np(NN, CC, NN)) --> nn(P1, NN), cc(P2, CC), nn(P3, NN), {P0 is
P1*P2*P3*0.200000}.
np(P0, np(PD, NN)) --> pd(P1, PD), nn(P2, NN), {P0 is P1*P2*0.200000}.
vp(P0, vp(VB, TA)) --> vb(P1, VB), ta(P2, TA), {P0 is P1*P2*1.000000}.
adj(1.000000, adj(شدید)) --> [شدید].
adv(1.000000, adv(فیصد)) --> [فیصد].
ca(1.000000, ca(80)) --> [80].
cc(1.000000, cc(اور)) --> [اور].
i(1.000000, i(بھی)) --> [بھی].
nn(0.166667, nn(لیے)) --> [لیے].
nn(0.166667, nn(بچوں)) --> [بچوں].
nn(0.166667, nn(تربیت)) --> [تربیت].
nn(0.166667, nn(چوٹیں)) --> [چوٹیں].
nn(0.166667, nn(درمیان)) --> [درمیان].
nn(0.166667, nn(دوڑ)) --> [دوڑ].
p(0.500000, p(کو)) --> [کو].
p(0.500000, p(کے)) --> [کے].
pd(1.000000, pd(اس)) --> [اس].
sm(1.000000, sm(.)) --> [.].
ta(1.000000, ta(ہیں)) --> [ہیں].
vb(1.000000, vb(آتی)) --> [آتی].
```

Fig. 4. A sample of PDCG format of sentence for execution in PROLOG

III. RESULTS

The results are evaluated through PARSEVAL measures, as mentioned in (Black et al. 1991). The precision, recall and crossing accuracy is calculated through following formulas, where M is the proposed parse and C is the correct parse in Gold standard.

$$Precision = \frac{\#\ correct\ parenthesized\ pairs\ in\ M}{\#\ parenthesized\ pairs\ in\ M} \qquad (2)$$

and similarly, the Recall is calculated as

$$Recall = \frac{\#\ correct\ parenthesized\ pairs\ in\ M}{\#\ parenthesized\ pairs\ in\ C} \qquad (3)$$

and the crossing accuracy is the percentage of parenthesis from M that does not cross over parenthesis in C.

The test data includes 2505 words while the others are used in training process which has been discussed earlier in the design section. The most likely parse given by SWI-PROLOG parser is compared with the manually annotated parse tree in the NU-FAST Treebank (The Gold Standard). The results are evaluated with lexicon and without lexicon. The precision, recall and accuracy measures with lexicon are given in Fig. 5 below.

Sentence Length	Average Length	Precision	Recall	Crossing Accuracy
2-12	7.4	92.8%	94.0%	98.5%
2-18	12.1	92.1%	93.4%	95.8%
2-24	17.2	91.2%	91.9%	94.2%
2-40	26.7	89.3%	88.5%	90.1%

Fig. 5. Precision, Recall and Accuracy with Lexicon

PARSEVAL measures are also evaluated without lexicon and found a little difference between the results. The SWI-PROLOG recursive parser outperforms with lexicon. The parsing results are given in Fig. 6 below.

Sentence Length	Average Length	Precision	Recall	Crossing Accuracy
2-12	7.2	90.3%	92.7%	98.2%
2-18	11.3	88.4%	92.2%	95.1%
2-24	16.8	86.8%	91.1%	93.3%
2-40	22.1	80.5%	82.8%	88.4%

Fig. 6. Precision, Recall and Accuracy without Lexicon

The result can not be compared with any existing research in Urdu, because it is the first step towards probabilistic parsing in Urdu. However, as compared to Eugene Charniak, it gives more accurate result with 1.7%, 1.0% and 0.3% for precision, recall and crossing accuracy in case of 2-12 sentence length and similarly, has an advantage over other lengths.

It is pertinent to note that during training process, the computation time for probability calculation of 22376 grammar rules is 18 hours and 48 minutes on 750 MHz Intel Processor with 384 RAM, which is no doubt a huge time. This process can be sped up using some high speed machine.

IV. ISSUES AND DISCUSSIONS

The poor results of parsing with untrained lexicon are also occurred due to SWI-PROLOG DCG based parser which stuck on occurring of mismatch of lexicon in PDCG. However, the tracer of SWI-PROLOG can show the progress at that stucking level. These mismatch like issues comes when a lexical entry is not present in training data, however, accuracy becomes a bit more improved if the lexicon matched in the PDCG. So, the issue is open, and can be improved if someone will take information from the tag if a lexicon is mismatched and the results will surely be improved with untrained lexicon too. In results presented in fig. 5 we ignore the untrained lexicons. However, with the help of SWI-PROLOG tracer, we have found the precision decrease almost 52% to 44% in length wise increasing order of sentences.

During manual annotation of trees, the tags of words like adverbs, adjectives, special nouns etc are always a hectic job, but accuracy depends on tags a lot. More informative tags can lead towards more accurate parsing results.

When the start symbol S occurred more than once in a sentence, its recursive behavior creates problems in parsing. Thisproblem can be solved by introducing some other sub-sentence feature tags like S1, or S2 or S-SUB to improve the accuracy.

Sometime, the correct parse tree of a sentence as per gold standard is present in parse trees, produced by the Prolog parser, but its probability is lower than the most likely parse generated by the parser. So, with an improved probability mechanism, one can generate those hidden correct parse trees or some other CYK or Early based parser can be used to fulfill this purpose.

Other issues like more accurate rules in grammar, error in gold standard etc. were also present to some extent.

CONCLUSIONS

The Tree-bank based grammar is no doubt, the best technique for building up the grammar for language through some easy and understandable steps as compared to theoretical modeling of a grammar for a language. Lexicon, proper tagging policy, efficient parsing technique, dynamic programming approach, more informative and division of phrases into further subdivisions may lead to efficient grammar. The NU-FAST tree-bank will provide a plate-form for probabilistic work in Urdu language processing.

REFERENCES

[1] An Official Document of Urdu Grammar Rules, 2005, Center for Research in Urdu Language Processing, National University of Computer & Emerging Science, Lahore, Pakistan (unpublished).

[2] Ann Bies, Mark Ferguson et al. 1995, Bracketing Guidelines for Treebank II style Penn Treebank Project.

[3] Christopher D. Manning, A Foundation of Statistical Natural Language Processing, The MIT Press, Cambridge, Massachusetts, London, England.

[4] Daniel Jurafsky and James H. Martin. 1999, Speech and Language Processing, Prentice Hall, Englewood Cliffs, New Jersey 07632.

[5] Eugene Charniak. 1996, Treebank Grammars, Department of Computer Science, Brown University, Rhod Island 02912.

[6] E. Black et al., 1991. A Procedure for Quantitatively Comparing the Syntactic Coverage of English. *Proceedings DARPA Speech and Natural Language Workshop*, Pacific Grove, Morgan Kaufmann.

[7] F.C.N. Pereira and D.H.D. Warren. 1980. Definite clause grammars for language analysis—a survey of the formalism and a comparison with augmented transitionnetworks. Artif. Intell., 13(3):231–278.

[8] Hassan Sajid, 2007, Urdu Part of Speech Tagset, Center for Research in Urdu Language Processing, National University of Computer & Emerging Science, Lahore, Pakistan (MS Thesis).

[9] Hindle, Donald, 1983. User manual for Fidditch, Technical memorandum 7590142, Naval Research Laboratory.

[10] Hindle, Donald, 1989. Acquiring disambiguation rules from text, In Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics.

[11] Mitchell Marcus, Grace Kim, et al.. 1994, The Penn Treebank: Annotating Predicate Argument Structure, Department of Computer and Information Science, University of Pennsylvania, USA.

[12] Mitchell P. Marcus, Beatrice Santorini, Marry Ann MarcinKiewicz, Building a large annotated corpus of English: The Penn Treebank, Penn Treebank Project (1989-1992).

[13] Santorini Beatrice and Marcinkiewicz Mary Ann, 1991. Bracketing guidelines for the Penn Treebank Project. Ms., Department of Computer and Information Science, University of Pennsylvania, USA

[14] Santorini Beatrice. June 1990, Part of Speech Tagging Guidelines for The Penn Treebank Project, 3rd Revision, 2nd Printing (February 1995),

[15] Tony Abou-Assaleh, Nick Cercone, Vlado Keselj, 2003, Expressing Probabilistic Context-Free Grammars In The Relaxed Unification Formalism, Faculty of Computer Science, Dalhousie University, Halifax, Nova Scotia, B3H 1W5, Canada.