STATISTICAL PARSER FOR URDU

by

Toqeer Ehsan
2015-Ph.D-CS-02

Research Supervisor:
Prof. Dr. Sarmad Hussain

2022

Department of Computer Science
University of Engineering and Technology, Lahore

STATISTICAL PARSER FOR URDU

by

TOQEER EHSAN

A DISSERTATION

presented to the university of engineering and technology, Lahore

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

APPROVED BY:

Dr. Sarmad Hussain
Professor, Al-Khawarizmi Institute of
Computer Science, UET, Lahore.

Prof. Dr. Mohammad Abid
Dean, Faculty of Numerical Sciences
and Information Technology,
City University of Science and Information
Technology, Peshawar.

Dr. Tafseer Ahmed Khan
Associate Professor, Department of
Computer Science,
Muhammad Ali Jinnah University,
Karachi.

Prof. Dr. Muhammad Shoaib
Chairman, Department of Computer Science

Prof. Dr. Muhammad Kamran
Dean, Faculty of Electrical
Engineering

June 06, 2022

UNIVERSITY OF ENGINEERING & TECHNOLOGY, LAHORE

ABSTRACT

A number of tools for Urdu language processing have been developed in the past few years to perform word segmentation, part of speech tagging, chunking, named entity recognition and parsing. For statistical analysis and modeling the linguistic features, annotated resources are essential. Corpora, especially treebanks, are required to perform statistical syntactic parsing. The parsing is helpful to depict the syntactic and semantic structure of a language. This research presents the development of a treebank for Urdu and its statistical parsing. The treebank has a multi-layered annotation scheme which includes part of speech tagging, phrase bracket labeling and functional labels. The syntactic annotation has been performed in The Penn Treebank style to mark phrases and grammatical functions. The treebank contains 7,854 annotated sentences and 148,575 tokens. To evaluate the annotation consistency, a grammar-based and automatic consistency checking based evaluations have been performed. The inter-annotator agreement is greater than 90 which was computed on reference corpus.

The parsing experiments have been performed by using different language representations including textual input, gold part of speech tags, lemmatized text and word clusters. In addition, several syntactic features have been experimented to analyze the parsing results. The feature set includes sub-categorization of part of speech tags, empirically learned vertical and horizontal markovizations and lexical phrase heads. For parsing experiments, probabilistic context-free grammars, data-oriented parsing and neural network based models have been trained on the Urdu treebank. We developed a bidirectional long-short term memory (BiLSTM) based parser and a POS tagger which have been trained on the final version of the treebank. Data-oriented and our BiLSTM parser performed with f-scores of 87.1 and 89.1 respectively. We further converted the treebank to the universal dependencies by devising head rules and dependency label mappings and performed dependency parsing. However, the dependency treebanking is an additional work of this dissertation.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my doctoral supervisor, Prof. Dr. Sarmad Hussain, whose guidance has provided a good basis for the present dissertation.

I express my deep gratitude to Prof. Dr. Miriam Butt for her supervision and guidance during my research stay at University of Konstanz. I am thankful for her detailed review, constructive criticism and excellent advice during this research.

I am thankful to Deutscher Akademischer Austauschdienst (German Academic Exchange Service) for providing me a funding opportunity to conduct my research at University of Konstanz, Germany. It helped me to enhance my knowledge and gave me an exposure of the research field.

STATEMENT OF ORGINALITY

It is stated that the research work presented in this dissertation consists of my own ideas and research work. The contributions and ideas from others have been duly acknowledged and cited in the dissertation. This complete dissertation is written by me. If at any time in the future, it is found that the thesis work is not my original work, the University has the right to cancel my degree.

Toqeer Ehsan

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

Symbol	Description
NLP	Natural Language Processing
PS	Phrase Structure
DS	Dependency Structure
POS	Part Of Speech
MRL	Morphologically Rich Language
PTB	Penn Treebank
UTB	Urdu Treebank
HUTB	Hindi Urdu Treebank
KON-TB	Konstanz Treebank
PCFG	Probabilistic Context-free Grammar
TSG	Tree Substitution Grammar
DOP	Data Oriented Parsing
RNN	Recursive Neural Network
LSTM	Long Short Term Memory
BiLSTM	Bi-directional LSTM
RFE	Relative Frequency Estimate
EWE	Equal Weight Estimate
CVG	Compositional Vector Grammar
Word2Vec	Words To Vectors
ELMo	Embeddings for Language Models

BIRA	Bidirectional Interpolated Refining Alternating
SOV	Subject Object Verb
OSV	Object Subject Verb
SVO	Subject Verb Object
NOM	Nominative
ERG	Ergative
ACC	Accusative
DAT	Dative
INS	Instrumental
ABL	Ablative
LOC	Locative
GEN	Genitive
SG	Singular
PL	Plural
MASC	Masculine
FEM	Feminine
Perf	Perfective
NN	Common Noun
NNP	Proper Noun
VBI	Main Verb Infinitive
VBF	Main Verb Finite
AUXA	Auxiliary Aspectual
AUXP	Auxiliary Progressive

AUXM	Auxiliary Modal
AUXT	Auxiliary Tense
PRP	Pronoun Personal
PDM	Pronoun Demonstrative
PRS	Pronoun Possessive
PRD	Pronoun Relative Demonstrative
PRR	Pronoun Relative Personal
PRF	Pronoun Reflexive
APNA	Pronoun Reflexive Apna
JJ	Adjective
Q	Quantifier
CD	Cardinal
OD	Ordinal
FR	Fraction
QM	Multiplicative
RB	Common Adverb
NEG	Negation
PRE	Preposition
PSP	Post Position
CC	Coordinate Conjunction
SC	Subordinate Conjunction
SCK	Subordinate Conjunction Kar
SCP	Subordinate Conjunction Pre-sentential

INJ	Interjection
PRT	Particle
VALA	Vala Particle
SYM	Symbol
PU	Punctuation
FF	Foreign Fragment
NP	Noun Phrase
VC	Verb Complex
PP	Postpositional Phrase
ADJP	Adjective Phrase
ADVP	Adverbial Phrase
QP	Quantifier Phrase
S	Simple Phrase
SBAR	Subordinate Clause
PREP	Prepositional Phrase
DMP	Demonstrative Phrase
FFP	Foreign Fragment Phrase
SUBJ	Subject
OBJ	Object
OBL	Oblique
ADJ	Adjunct
POF	Part Of Function
PDL	Predicate Link

VALA	Vala Verb
VOC	Vocative
INJ	Interjection
G	Genitive
NLU	Natural Language Understanding
LR	Labeled Recall
LP	Labeled Precision
LAS	Labeled Attachment Score
UAS	Unlabeled Attachment Score
LA	Labeled Accuracy

1. INTRODUCTION

Urdu is rich in morphology and is written by using a version of the Arabic script from right to left. It is an Indo-Aryan language which is mainly spoken in the South Asian region. There are over 160 million speakers who speak Urdu as first and second language all over the world [138]. It has several distinguishing linguistic properties which include rich morphology [79], case system [26, 85], a structure of complex predicates [28, 23] and flexible word-order [126]. These characteristics of the language make it more challenging to perform the tasks of language processing. It is considered as low resourced language, however a number of efforts are undertaken in the past decade and several essential linguistic resources have been developed to perform NLP tasks. These tasks include development of Urdu corpora, analysis of morphology, tokenization and word segmentation, part of speech tagging, sequence chunking and syntactic parsing. Urdu remains under-resourced despite these efforts and there is a need to develop linguistic resources and tools to achieve competitive computational results as compared to computationally developed languages.

The corpus-based methods are more focus by the modern research for language processing tasks. Treebanks are essential resources for a range of NLP tasks. A number of treebanks have been developed by using phrase structure (PS) [107, 103, 152, 116,

117] and dependency structure (DS) [100, 72, 22, 136]. The development of Penn Treebank gave new perspectives and inspirations for the development of syntactic resources for many other languages and corpus based syntactic analysis. The Penn Treebank (PTB) was a central resource during the development of statistical parsers. PTB has a phrase structure annotation scheme along with functional labels to represent grammatical roles. The phrase structure annotation scheme was influential for the development of resources of many other languages. Dependency structure is another annotation method which represents the grammatical relations by using dependency labels and shows dependencies on head words. Currently, a lot of treebanks have been developed having dependency structures.

To perform the task of statistical parsing, an Urdu treebank is required with sufficient number of annotated sentences. Therefore we have developed a treebank for Urdu (CLE-UTB) which has been annotated by using phrase structure annotation and performed constituency parsing. The treebank was later converted to dependency structure to perform dependency parsing. The CLE-UTB was initiated to investigate the relationship between syntax and prosody of an annotated speech corpus of Urdu. However, the size of the treebank was extended to have 7,854 sentences and an extensive evaluation was undertaken to produce a consistent resource. Prominent Urdu treebanks which existed already are Hindi-Urdu Treebank (HUTB) [14] and the Urdu.KON-TB [2]. The HUTB uses the Paninian grammatical model [11] to annotate dependency structured trees. The HUTB guidelines for phrase structure have been derived from the Minimalist Program [33]. It proposed binary trees and has specific positions for syntactic arguments which enforces the complex tree structure for constituents with

flexible word order. The focus of this research is on syntactic structure of Urdu language only. By adding more information like morphological features in the annotation process will increase the data requirements significantly which would make it more challenging task for a low-resourced language Urdu.

In contrast, a phrase structure annotation scheme has been used for the development of the Urdu.KON-TB. The annotation scheme has a composite label set. The annotation labels attach the information about case, tense, aspect and modality with them. It has a large annotation scheme containing 26 phrase labels. Although, it is a small sized resource containing 1,400 sentence with an average length of 13.73 words. Because of its complex annotation scheme and smaller size, we initiated to develop a larger treebank for Urdu. The annotation scheme of our treebank provides a simplified and consistent annotations. The guidelines for syntactic annotation are inspired by the Urdu.KON-TB with few additional structures. However, the annotation scheme for the development have been derived from several resources including a universal tag set [73], Penn Treebank and HUTB. Our annotation scheme provides several advantages in the process. Less complex parse tree are produced due to the simplicity of the tag set. The scheme is in-line with other resources rather than having a totally new label set. The scheme helps to minimize the data requirements to train statistical parsing models. Alongside the smaller tag set, our annotation scheme is quite appropriate to annotate constituency structure of Urdu. Chapter 3 discusses the annotation label set and annotation guidelines of the CLE-UTB in more detail.

The development of a consistent linguistic resource is essential for corpus based NLP to attain valid results. We have performed multiple evaluation tasks during the

annotation of the treebank. These tasks include completeness and correctness checking, a grammar-based evaluation and automatic treebank consistency checking. In the completeness and correctness checking, we ensured that all the tokens and phrases have a correct label. The checking has been done for POS tags, phrase and functional labeling. Under the grammar-based evaluation, potentially implausible grammar rules have been identified. A grammar was extracted containing those rules and the reported sentences were reviewed according to the annotation guidelines. Finally, an automatic consistency checking took has been used to identify incorrect annotations according to their contexts. The tool has been employed to evaluate POS tags, phrase and functional labels. It identified outliers which were further reviewed manually. Before performing grammar-based evaluation and consistency checking, a random reference corpus was extracted to perform inter-annotator agreement. The overall agreement was higher than 90% for phrase label annotations. After treebank evaluation, we performed the treebank analysis with respect to text genres and reported statistics of the labeling in the treebank. For the annotation of the treebank a balanced corpus has been collected which contains text from fifteen different text domains. The developed treebank contains 7,854 sentences with 148,775 tokens.

This work presents the state-of-the-art results by performing constituency parsing for the CLE-UTB. To parse morphologically rich and relatively free word-order languages like Urdu is not a trivial task because they have diverse vocabulary with respect to surface forms. The flexible word order also produces a number of combinations of phrases in a treebank which increases the data requirements to train statistical

models. Several parsing techniques and formal grammars are available for natural language parsing. In this work, we have done experiments with statistical models which include probabilistic context-free grammars, data-oriented parsing (DOP), lexicalized grammars, a recursive neural network based parser and a bi-directional long-short term memory (BiLSTM) model to parse our treebank. We have developed a parser and POS tagger to train on the CLE-UTB which are based on BiLSTM networks. The results were further enhanced by performing transfer learning by using word representations which were trained on a large Urdu corpus. Our BiLSTM parser and POS tagger outperformed the grammar based parsing models regardless of employment of linguistic features with them. Before training neural parser, several linguistics properties and features were used to enhance the parsing results for grammar-based models. These features were helpful to increase parsing scores by improving learning capabilities of the parsers. The linguistic features included dependency knowledge of case markers of Urdu and POS sub-categorization on the bases of that information, head-word for lexicalized parsing and phrase parental annotation in parser trees, lemmatization and word clusters.

The statistical parsing of morphologically-rich languages (MRLs) prompts specific challenges to be catered first as examined by [148]. Tsarfaty *et al.* [148] brought up three essential issues which are needed to be offered an explanation to parse an MRL. 1) What is the type of input and language representation? 2) How should morphological data be represented at the annotation step? 3) How much training data is required for preparing a parser? The first question asks about the form of input to the parser whether

it is plain text, lemmatized text, text with their POS tags or some other encoding or clustering method which could be helpful for better parsing results. The second question is about the annotation scheme and implementation of the morphological information at phrase level. The annotation scheme is influential to answer the third question. A larger and morphologically richer annotation labeling will lead to have more training data. This work also responds to these inquiries and performs phrase structured parsing for Urdu. For the sake of simplicity of meaningfulness in this thesis, all the examples are exhibited by employing a Roman transliteration scheme proposed by [105].

The parsers have been assessed against a solitary test set which was further divided into three categories; small, medium and long sets. The test set was prepared in a way that it contains text from all genres and sentence length of the corpus. The evaluation has been performed on plain Urdu text as well as by encoding POS labels with the tokens.

The dependency based POS sub-categorization for post-positions and punctuation symbols were helpful to improve the performance of grammar-based parsers. Therefore, the later models have been trained and tested on the extended POS tag set. The lemmatization was further performed to diminish the information sparsity and it produced an unobtrusive improvement in the parsing scores for lexicalized PCFG parsing model. An unsupervised word clustering method categorized the tokens into groups depending on their grammatical similarities. We performed lexicalized parsing by replacing tokens with their cluster labels. For word clustering, we have used a predictive exchange algorithm as described in [48]. The word clustering was helpful to enhance the parsing results. Urdu has adaptable word-order, in this way the DOP and RNN

models have been additionally assessed on a test set which was classified with various orders. However, our BiLSTM parser outperformed other parsers by employing transfer learning.

The dependency structure produces the annotation of grammatical information by showing dependencies on head-words and dependency labels to represent their relationships. The annotation scheme of the CLE-UTB is compatible with dependency structure as it uses a relatively flat structure which is helpful to mark flexible word order of the language. The phrase labels are encoded with functional tags which can be mapped on dependency labels to represent dependency relationships. We have converted our treebank to dependency structure to perform dependency parsing. However, the dependency treebank and parsing is additional work of this dissertation and is still in progress. Therefore the detail of phrase to dependency conversion and dependency parsing are presented in Appendix A. We have derived a head-word model and a phrase to dependency label mappings. The head-word model identifies the head words with in constituents and for dependency label mapping, we have used the Universal Dependency version 2.0 (UD 2.0)[1] label set. Several post conversion rules have been employed to enhance the conversion accuracy. We have further trained the dependency parsers on the converted dependency treebank and compared the results with an existing Urdu dependency treebank which is a part of HUTB. Our treebank produces competitive dependency parsing results when trained on a BiLSTM parser[2].

The syntactic parsing is useful for a number of language processing tasks including sentiment analysis, semantic parsing, machine translation, speech recognition

[1] https://universaldependencies.org/
[2] https://github.com/elikip/bist-parser

and text to speech systems. A phrase based sentiment classification has been performed for Brazilian Portuguese in [43]. They have performed lexicon-based analysis by incorporating phrase structure based on X-bar theory. The sentiments were computed for all the constituents which were further combined to assign a polarity to the whole sentence. Similarly, Chinese language sentiment analysis has been performed for Weibo[3] in [97]. They have trained latent Dirichlet allocation (LDA) model along with Gibbs sampling for inference. They further used the dependency structure information to find the details of subjects and objects. The syntactic parsing has been performed by using a parser presented in [98]. Beside incorporating syntactic parsing in sentiment analysis, an annotated corpus has been developed on a phrase structure treebank [142]. The corpus has been annotated with sentiment labels along with syntactic categories. The parse trees have been achieved by using Stanford parser [94].

The syntactic parsing is also helpful for the semantic annotation and parsing by providing the grammatical relations of phrases and lexical items. A semantic parsing has been performed by using syntactic parsing in [156]. They proposed a neural network based joint model to perform semantic and syntactic parsing. They used semantic details to perform phrase structure and dependency parsing and vice versa. They reported promising results as compared to existing models. Another semantic analysis for user utterances of task oriented dialog has been proposed in [71]. They further built a semantic dataset which used a constituency parser to attain the hierarchical structure of the sentences. The constituency parsing has been further used to parse sentences with elided elements [83]. Semantic roles have been used to identify gapping elements.

[3] www.weibo.com

The syntactic parsing has been further applied in machine translation and speech processing systems. A hybrid decoder for neural machine translation has been proposed by employing recurrent neural network grammars (RNNG) [53] [56]. The hybrid model has been combined by joining the language model decoder and RNNG while sharing word vectors. Another method to incorporate parse tree structure in the neural machine translation has been proposed in [151]. The model generates the tree structure on target side and the parser is similar to the one presented in [53]. The syntactic information has also been used for automatic speech recognition [42]. The syntactic parsing has been used in speech synthesis in [70]. The have improved the quality of pronunciation, prosody and naturalness in the produced speech by using the syntactic categories from parse trees of the sentences. The phrase structure has been used for word relations and this information was helpful to improve the system performance.

The thesis has been organized as follows. Chapter 2 presents the related works and review of literature for treebank development and statistical parsing. Chapter 3 discusses the development of the treebank CLE-UTB. It presents the details of the collected corpus, annotation scheme, annotation guidelines, treebank evaluation steps and phrase to dependency structure. Chapter 4 presents the details of the parsing models and linguistic features. The parsing models include grammar-based parsers, data-oriented parsing, lexicalized grammars, recursive neural network parser and the proposed BiLSTM parser. Chapter 5 presents the constituency and dependency parsing and POS tagging results and discussions. Appendix A discusses the dependency treebank and parsing. Chapter 6 provides the conclusions and discusses future works.

2. LITERATURE REVIEW

This chapter presents the detailed survey of the related works which have been influential for the development of this computational resource for Urdu. Several prerequisites are required to perform syntactic analysis and annotation of any language. The language representations, in terms of character sets and lexical analysis provide distinctive units for structural analysis. In case of Urdu, word segmentation, morphological richness and free word order pose fundamental challenges at the first step. The second step is about the selection of suitable grammatical structure and its annotation scheme. An annotation scheme should be compatible with the related resources and it should be able to annotate the syntax of a language. Correct and consistent annotation of a resource is important to train and evaluate machine learning algorithms. In this chapter, we present the related works starting from fundamental tasks to the statistical parsing of morphologically rich languages

This chapter is organized as follows. Section 2.1 presents the basic characteristics of Urdu. Section 2.2 presents the literature for different types of treebanks and their annotations. This section includes discussions for phrase structures as well as dependency structure treebanks. Section 2.3 describes parsing techniques and their evaluations for different languages. Section 2.4 presents the literature about parsing morphologically-rich languages.

2.1 CHARACTERISTICS OF URDU

This section presents basic characteristics and fundamental challenges of Urdu. Following sections describe word segmentation, morphological richness, case system, free word order, complex predicates and phrasal heads. All these properties are necessary to perform lexical and syntactic analysis of Urdu.

2.1.1 Word Segmentation

Urdu uses an extended version of Arabic script which is a cursive script and is written from right to left. Several Unicode based phonetic keyboard are available to type Urdu on computers. The Arabic Unicode chart [1] contains characters for many languages which are written by using a subset of the whole script. Characters have different properties which change their shapes when used with other characters. A single character can be written at four different positions in a word which include initial, middle, final and isolated positions [52]. One or more characters combine to make ligatures and ligatures further constituent words. An isolated character can appear as a ligature in a word. The characters which combine to make ligatures, usually change their shapes depending on their positions.

Joining is the most important property of a character in the Arabic script. Some characters are two sided joiners, that is, they can appear at initial, middle and final positions of a ligature. These alphabets include 'Bay','Pay','Jeem' etc. Some characters are one sided joiners which means they only appear at the final position of ligatures. These letters include 'Ray', 'Daal', 'Wao' etc. Any letter appearing in isolated form at final

[1] https://www.unicode.org/charts/PDF/U0600.pdf

position of the previous letter, is a one sided joiner. The letter 'Bari-Yey' always appears at the end of ligatures or in isolated form. It does not appear at initial or middle position of any ligature. The joining properties of the script, produce words with various number of ligatures. The characters may have different shapes by appearing at different positions. Spaces are usually used to separate ligature and to maintain the plausible shapes but not to separate words essentially. The ligatures and words appear separated to the reader even if the space character is missing. Therefore, the writing system has a basic problem of word segmentation. For the word and phrase level computational analysis, the tokenization is fundamental task which is not directly supported by the script.

To compute the language, one should perform manual tokenization or implement an automated word segmentation. A few word segmentation systems have been developed for Urdu [52, 8, 18]. In [52], the space errors have been divided into two categories, space omission errors and space insertion errors. Figure 2.1 shows an example of space omission errors.

(a) لڑکےنےشیرکودیکھا
(b) لڑکے نے شیر کو دیکھا

laRkE=nE SEr=kO dEkHA
boy.M.Sg.Erg lion.M.Sg.Acc see.Perf.Sg
'The boy saw the lion.'

FIGURE 2.1: Space omission errors. (a) Incorrect: no spaces between words. (b) Correct: space after each word.

Figure 2.1 (a) does not have any spaces between words of the sentence. The words show plausible shapes to readers because of non-joiner characters at the final position of each word. However, the tokenization of such sentences and words create erroneous word by combining more than one lexical items. These types of errors are

referred are space omission errors [52]. Figure 2.2 shows examples of space insertions errors where spaces are inserted between words to keep the shapes of the words.

(a) ذمہ داری (Responsibility)
(b) خوش مزاج (Cheerful)
(c) جنتر منتر (Mantra)
(d) ٹیلی وژن (Television)
(e) یو ای ٹی (U.E.T)

FIGURE 2.2: Space insertion errors. (a) zimE dArI (Responsibility) (b) xuS mizAj (Cheerful) (c) jantar mantar (Mantra) (d) TElI vIyan (Television) (e) U.E.T (Abbreviation)

Figure 2.2 shows five space insertion issues where spaces have been typed between words. The space insertion errors have been divided into five categories which include space insertions for affixation, compounding, reduplication, foreign words which are transcribed into Urdu and abbreviations [52].

The automatic word segmentation systems have been developed to overcome these errors automatically [52, 8, 18]. An n-gram based ranking algorithm has been developed which achieved an overall word segmentation accuracy of 95.8% [52]. The segmentation accuracy was further improved by using ligatures are basic units rather than words. The ligature based n-gram model achieved the best accuracy of 96.1% [8]. Another Urdu word segmentation system was developed by applying conditional random field in [18]. They have used manually annotated data to predict space as word boundary and zero-width-non-joiner (ZWNJ) as sub-word boundary where it was used to overcome space insertion errors. The model achieved an F_1 score of 97% for space omission errors and 85% for space insertion errors. In our research, we developed a treebank by using manual annotation of sentences. Automated word segmentation may not produce hundred percent accuracy therefore, we have segmented the corpus manually during the annotation. The space omission problems have been solved by

inserting spaces between words and for space insertion errors, ZWNJ character has been used to mark sub-word boundaries.

2.1.2 Morphological Richness

A morphologically rich language (MRL) has many surface forms for a single root word. Urdu has more than 25 forms for a verb. It has same number of additional forms for causative verbs and double causatives. There are a total seventy five forms for a verb [79]. An Urdu verb has surface forms for tense, number, gender, person and honorifics. Table 2.1 shows the forms of an Urdu verb کھا 'eat' along with its morphological features.

Table 2.1 shows 25 underlying forms against non-causative, causative and double causative surface forms. There are total 75 underlying forms and 51 unique surface forms. The underlying forms represent the morphological properties with respect to tense, number, gender, person and honorifics. The underlying forms are divided into five categories including infinitive, past, habitual, non-past and commands.

In case of nouns, there are forms for number, gender and case. Urdu has its own vocabulary and morphology however, in some cases, it borrows the morphology from Arabic and Persian. Therefore, there are many affixation rules for Urdu nouns [79]. These include morphological rules for thesw words from Persian and Arabic as well. There are different rules for masculine and feminine nouns. Table 2.2 presents examples of simple masculine and feminine nouns and their morphological forms.

Table 2.2 shows surface forms along with underling forms of two nouns. Urdu has a diverse morphology hence has many morphological rules for making surface

TABLE 2.1: Surface forms of an Urdu verb including causative and double causative forms.

No.	Features	Non-Causative	Causative	Double Causative
1	Root	کھا	کھلا	کھلوا
2	Infinitive singular	کھانا	کھلانا	کھلوانا
3	Infinitive feminine	کھانی	کھلانی	کھلوانی
4	Infinitive plural	کھانۓ	کھلانۓ	کھلوانۓ
5	Past masculine singular	کھایا	کھلایا	کھلوایا
6	Past feminine singular	کھائی	کھلائی	کھلوائی
7	Past masculine plural	کھائے	کھلائے	کھلوائے
8	Past feminine plural	کھائیں	کھلائیں	کھلوائیں
9	Habitual masculine singular	کھاتا	کھلاتا	کھلواتا
10	Habitual masculine plural	کھاتۓ	کھلاتۓ	کھلواتۓ
11	Habitual feminine singular	کھاتی	کھلاتی	کھلواتی
12	Habitual feminine plural	کھاتیں	کھلاتیں	کھلواتیں
13	Non-past 3P singular	کھائے	کھلائے	کھلوائے
14	Non-past 3P plural	کھائیں	کھلائیں	کھلوائیں
15	Non-past 2P singular	کھائے	کھلائے	کھلوائے
16	Non-past 2P plural honor-1	کھاؤ	کھلاؤ	کھلواؤ
17	Non-past 2P plural honor-2	کھائیں	کھلائیں	کھلوائیں
18	Non-past 2P plural honor-3	کھائے	کھلائے	کھلوائے
19	Non-past 1P singular	کھاؤں	کھلاؤں	کھلواؤں
20	Non-past 1P plural	کھائیں	کھلائیں	کھلوائیں
21	Command singular	کھا	کھلا	کھلوا
22	Command plural honor-1	کھاؤ	کھلاؤ	کھلواؤ
23	Command plural honor-1	کھائیو	کھلائیو	کھلوائیو
24	Command plural honor-2	کھائیں	کھلائیں	کھلوائیں
25	Command plural honor-3	کھائۓ	کھلائۓ	کھلوائۓ

forms. Simple affixation rules are applied to these nouns. The forms have been shown for nominative singular and plural nouns. Other forms have been achieved against ergative, accusative and vocative cases.

Different feature annotation schemes are available to use in the development of datasets. The parallel grammar project (ParGram) provides a framework to perform grammatical analysis of a language by writing constituency and functional structure of a sentence [29, 25]. It also provides an annotation scheme for morphological features.

TABLE 2.2: Surface forms of Urdu nouns.

No.	Features	Root	Surface Form
1	Nominative masculine singular	لڑکا	لڑکا
2	Nominative masculine plural	لڑکا	لڑکے
3	Ergative masculine singular	لڑکا	لڑکے
4	Ergative masculine plural	لڑکا	لڑکوں
5	Accusative masculine singular	لڑکا	لڑکے
6	Accusative masculine plural	لڑکا	لڑکوں
7	Vocative masculine singular	لڑکا	لڑکے
8	Vocative masculine plural	لڑکا	لڑکو
1	Nominative feminine singular	لڑکی	لڑکی
2	Nominative feminine plural	لڑکی	لڑکیاں
3	Ergative feminine singular	لڑکی	لڑکی
4	Ergative feminine plural	لڑکی	لڑلیوں
5	Accusative feminine singular	لڑکی	لڑکی
6	Accusative feminine plural	لڑکی	لڑکیوں
7	Vocative feminine singular	لڑکی	لڑکی
8	Vocative feminine plural	لڑکی	لڑکیو

Universal Dependencies V2.0 provide a tag set for the annotation of morphological features[2]. It produces labels for lexical and inflectional features for nouns and verbs. Similarly, Universal Morphology (UniMorph) project provides a framework to annotate morphology of any language [90]. Currently, there are data sets for more than one hundred languages including Urdu [3].

Morphological richness of a language requires large amount of text to produce computational resources as it may cause data sparsity for machine learning algorithms. The annotation labels at lexical and phrase level need to be devised to take care of this issue. Beside the morphology of single word, Urdu also produces forms with multiple lexical items from affixations, compounding, reduplications, transcription of foreign

[2]https://universaldependencies.org/u/feat/index.html
[3]https://unimorph.github.io/

words and abbreviations. The morphological phenomena also create the issue of word segmentation. The second question asked in [148] is about the influent of morphology when designing an annotation scheme to develop a treebank. Section 3.1 describes the design of our POS and phrase label sets in more detail.

2.1.3 Case System

Cases represent the grammatical roles of nouns, adjectives, pronouns or numerals in the sentences. Urdu reflects its case systems by using clitics. These clictics are also called case markers. The case markers appear as isolated lexical items in the phrases hence attain a lexical category. They usually appear after the main word classes like nouns, pronouns, adjectives etc. Urdu has eight cases [85, 24, 26] which are shown in Table 2.3.

TABLE 2.3: Urdu case markers.

Sr#	Case	Clitic	Grammatical Function
1	Nominative	none	Subject/Object
2	Ergative	nE	Subject
3	Accusative	kO	Object
4	Dative	kO	Subject/Indirect Object
5	Instrumental	sE	Oblique/Adjunct
6	Ablative	sE	Oblique/Adjunct
7	Genitive	kA/kI/kE	Subject/Object specifier
8	Locative	mEN/par/tak	Oblique/Adjunct

Table 2.3 shows case markers, clitics and their grammatical functions. The nominative case does not use any clitic but it is used to represent subjects and objects. Ergative and accusative cases use clistics ‘nE’ and ‘kO’ to mark subjects and objects. The

sentence in (1) below shows an example of nominative case. The example in (2) presents ergative and accusative cases.

(1) mOminA skUl jAtI hE

Momina.F.Sg.Nom School.M.Sg.Nom go.Pres.F.Sg be.Pres.3.Sg

'Momina goes to school.'

(2) mOminA=nE sAim=kO bulAyA

Momina.F.Sg=Erg Saim.M.Sg=Acc call.Perf.M.Sg

'Momina called Saim.'

The dative case uses the 'kO' clitic to mark indirect object as shown by (3). Dative cases are used to mark dative subjects and indirect objects.

(3) sAim=nE MOmina=kO kitAb dI

Saim.M.Sg=Erg Momina.F.Sg=Dat book.Nom.F.Sg give.Pres.3.F.Sg

'Saim gave the book to Momina.'

Example (4) presents the instrumental case which uses 'sE' clitic. Instrumental cases usually mark oblique or adjunct grammatical roles in the sentences.

(4) mOminA pensil=sE acHA likHtI hE

Momina.F.Sg.Nom pencil.F.Sg=Ins good.M.Sg Write.Pres.F.Sg be.Pres.3.Sg

'Momina writes better with a pencil.'

Example (5) demonstrates genitive case by using the clitic ‘kA’. According to [26], genitive case markers are used as specifiers for subjects in infinitive clauses, finite copula constructions and simple nominal specifiers. The most common use is the nominal specifier to show possessions.

(5) kEf=kA bastA

Kaif.M.Sg=Gen bag.M.Sg

‘Kaif’s bag.’

Example (6) shows a locative case by using the case marker ‘par’. Locative case markers usually mark obliques for compulsory arguments and adjuncts for nor-core arguments.

(6) kitAb mEz=par rakHI hE

Book.F.Sg.Nom table.M.Sg=Loc put.Perf.F.Sg be.Pres.3.Sg

‘The book is on the table’

Understanding of the case system is important for the grammatical analysis of Urdu. The examples discussed above present the contribution of different cases to mark grammatical relations. In Urdu, these case markers appear as independent tokens demanding a lexical category. Therefore, the lexical and phrasal level annotation should have labels to label them. Chapter 3 describes label sets for the annotation of the Urdu treebank and annotation guidelines.

2.1.4 Free Word Order

The word-order analysis is crucial for syntactic annotation of the argument structure. Urdu has a flexible word order [126]. The grammar usually has a subject-object-verb (SOV) order but other orders also appear in the corpus like SVO, OSV etc. Adjunct arguments further have flexible order in the sentences. The positions of arguments are flexible but within the phrases, words and clitics keep a fixed order. Table 2.4 demonstrates the flexible order property by showing all possible orders of arguments for a sentence.

TABLE 2.4: Different word orders of a single sentence.

laRkE=nE	SEr=kO	dEkHA
boy.M.Sg=Erg	lion.M.Sg=Acc	see.Perf.M.Sg
'The boy saw the lion.'		

1. *laRkE=nE* (the boy) *SEr=kO* (the lion) *dEkHA* (saw)
2. *SEr=kO* (the lion) *laRkE=nE* (the boy) *dEkHA* (saw)
3. *SEr=kO* (the lion) *dEkHA* (saw) *laRkE=nE* (the boy)
4. *laRkE=nE* (the boy) *dEkHA* (saw) *SEr=kO* (the lion)
5. *dEkHA* (saw) *laRkE=nE* (the boy) *SEr=kO* (the lion)
6. *dEkHA* (saw) *SEr=kO* (the lion) *laRkE=nE* (the boy)

The sentence in Table 2.4 has three arguments, an ergative subject, an accusative object and a verbal structure. All three core arguments can appear in any order resulting in plausible argument structure. All the given orders have the ability to convey correct meaning of the sentence. However, the syntactic annotation of different orders would show parse trees with different shapes.

The Urdu word order is also effected by *ezafe* constructions which are mainly borrowed from Persian The *ezafe* constructions constitute noun phrases with reverse

order of nouns and their modifiers [21]. A noun phrase normally contains one or more modifiers followed by a noun. In *ezafe* constructions, a noun appears before its modifier. These constructions use a vowel sound which is pronounced with head word appearing at the start of the constituent. For example, a noun phrase *gul=e tAzA* 'fresh flower' has a noun modifier *tAzA* 'fresh' appearing after the noun while a clitic 'e' is pronounced with noun *gul* 'flower'. The *ezafe* clitic is represented by using a diacritic but it is not essential part of the script. A similar construction is used to show possession in a noun phrase. In this case, possession is associated to the noun on right while the noun on left is the head. For example, *hakUmt=e Pakistan* 'The government of Pakistan' shows possession by using 'e' with a changed word order.

2.1.5 Complex Predicates

Urdu also uses complex predicate structure to depict the verbal concepts. The complex predicates make combinations such as verb+verb, noun+verb, adjective+verb and quantifier+verb [23, 112, 30, 7]. For the complex predication with nouns, adjectives and quantifiers, the first portion contains the core predicate which are usually followed by light verbs. The auxiliary verbs further appear after light verbs with respect to tense, aspect and modality of the sentences. According to [30] and [7], frequent light verbs are *kar* 'do', *hO* 'be', and *dE* 'give'. However, there are also other verbs which are used with specific combinations. In (7), the constituent *SurU kI* 'started' makes noun+verb complex predicate structure. The noun *SurU* 'start' produces the semantics of the action and the light verb *kI* 'do.Perf.F.Sg' further provides the information of tense, gender and number.

(7) laRkE=nE paRHAI SuRU kI

boy.M.Sg=Erg study.F.Sg.Nom start.Sg do.Perf.F.Sg

'The boy started the study.'

The example presented in (8), show the adjective+verb structure. The constituent *band kar diyA* 'closed' has two components, and adjective *band* 'close' and a verbal structure *kar diyA* 'done'. The verbal structure, in this example, contains a light verb followed by an aspectual auxiliary verb.

(8) sAim=nE darvAzA band kar diyA

Saim.M.Sg=Erg door.M.Sg.Nom close.Sg do.Imperf.Sg give.Perf.M.Sg

'Saim closed the door.'

The example in (9) presents the complex predicate structure containing quantifier+verb. The quantifiers are usually handled as noun modifiers but our annotation marks it by using a separate label. The lexical sequence *kam kar dI* 'reduced' represents the complex predicate structure including a quantifier *kam* 'less' and the verbal structure *kar dI* 'done'. The aspectual auxiliary verb *dI* 'give.Perf.F.Sg' appears after the light verb.

(9) intizAmiyA=nE fIs kam kar dI

administration.F.Sg=Erg fee.F.Sg.Nom less.Sg do.Sg give.Perf.F.Sg

'The administration reduced the fee.'

Urdu also have complex aspectual structures which have been divided into three types [28]. The type-1 of the complex aspectual structures uses a verb+verb (V1+V2) structure. The V1 in the construction usually appears in the root form and the second verb V2 shows the information of aspect, tense, gender and number. Example (10) shows type-1 construction by using the construction *kHA liyA* 'ate'. *kHA* 'eat' is the main verb and *liyA* 'take.Perf.M.Sg' defines the aspect.

(10) mOminA=nE sEb kHA liyA

Momina.F.Sg=Erg apple.M.Sg.Nom eat.Sg take.Perf.M.Sg

'Momina ate the apple.'

The type-2 complex aspectual construction contains infinitive oblique verb in combination with another verb. In (11), the infinitive verb *banAnE* 'to make' is followed by a finite verb *lagI* 'began'. The type-1 construction is the part of a single constituent but the type-2 is annotated by constructing the non-finite clause in combination of the verbal construction.

(11) vO cAE banAnE lagI

pron.Sg.Nom tea.F.Sg.Nom make.Inf.Obl be.attached.Perf.F.Sg

'She began to make tea.'

The type-3 of complex aspectual structure has cases involved with the infinitive verbs which are further combined with the second verb. In (12), the construction *sEb kHAnE=kO* 'to eat the apple' defines the dative case for a non-finite clause as an oblique argument.

(12) mOminA=nE sAim=kO [sEb kHAnE]=kO

Momina.F.Sg=Erg Saim.M.Sg=Dat apple.M.Sg.Nom eat.Inf.Obl

kahA

say.Perf.M.Sg

'Momina told Saim to eat the apple.'

The positions of nouns, adjective, quantifiers, non-finite constructions are usually pre-verbal but this is not the case always. Urdu has a flexible word order therefore the discussed constructions may not always co-occur with verbal structures. To annotate such constructions, we need a mechanism which efficiently represents the flexible order of the language.

2.1.6 Phrasal Heads

The phrase structure parse trees represent the constituents in the hierarchical way. The constituents are identified by the phrase labels. They have multiple lexical items which may have different syntactic and semantic relations with each other. For example, a canonical noun phrase contains the main noun along with noun modifiers. The noun modifiers are dependent on the noun whereas the main noun actually defines the constituent to become a noun phrase. Similarly, other constituents are defined on the bases of their lexical heads. The phrasal heads are important to perform accurate constituency parsing. For English constituency parsing, the first head word method was proposed in [104]. The head model was further employed to perform lexical conditioning in the parsing models by [36, 38]. A head model for Urdu has also improved the

constituency parsing scores. Table 2.5 presents the Urdu phrasal heads with respect to our phrase label set. The phrasal head model has been described in Section 4.3.3 with more detail.

TABLE 2.5: Phrasal head for the Urdu treebank.

Phrase Label	Direction	Priority
VC	left to right	VBF, VBI, AUXA, AUXM, AUXP, AUXT, VC, NEG
PP	left to right	NP, S, QP, NNP, NN, PP, PSP
NP	right to left	NP, NNP, NN, PRP, PRR, S
ADJP	right to left	ADJP, JJ, Q, QP, RB
QP	right to left	QP, Q, CD, OD, FR, QM, JJ
ADVP	right to left	ADVP, RB, NP, NN
PREP	right to left	NP, NNP, NNP, PREP
DMP	right to left	PDM, PRP, PRT
FFP	left to right	FF, NNP, NN
S	left to right	VC, S, SBAR, NP, ADJP, QP NNP, NN, PRP,
SBAR	left to right	S, SBAR, SCK

Table 2.5 has three columns, first column shows the phrase labels, second column describe the direction of the head search against the phrase labels. The third column presents the label priority. The left-hand side label has the highest priority shown in the third column. For example, the VC (verb complex) phrase is left headed and the highest priority tag is VBF followed by VBI and POS tags for auxiliaries and negative tokens. A finite verbal structure, contains the main verb having tag VBF. The head model will identify the main verb as phrasal head. Similarly, the PP (post-positional) phrase is left headed as it contains the inner noun phrase which is usually annotated on

the left-hand side followed by case markers. Our phrase structure treebank has been annotated by using 11 phrase labels which have been described in Section 3.1.

2.2 TREEBANKS

A treebank is an essential resource to perform statistical syntactic parsing. A treebank contains the collection of annotated sentences. There are several annotation methods but the well-known representations are phrase structure and dependency structure. In this section, we describe phrase and dependency structure treebank annotations and their properties.

2.2.1 Phrase Structure Treebanks

A phrase structure annotation provides the hierarchical attachment of the constituents for a sentences. Phrase labels are used to identify specific constituents. The functional label are usually attached on the phrase labels to mark the grammatical relations of the constituents in parse trees. A typical phrase structure parse tree has multiple annotation layers including, word segmentation, part of speech (POS) tagging, phrase labels and a functional layer. One of the prominent phrase structure treebanks was developed for English called The Penn Treebank (PTB) [107]. The PTB was initially annotated to have the layers of POS tags and phrase labels. It was further updated to have an additional layer of functional tags to annotate the predicate argument structure [106]. The treebank uses 36 main POS tags and 12 additional tags for punctuation and symbols. The phrase labels were further extended to have 26 phrase labels and 20 functional labels [16]. Figure 2.3 shows an example sentence by using the annotation of the

PTB. Table 2.6 presents the phrase label set and Table 2.7 shows the functional labels of the PTB.

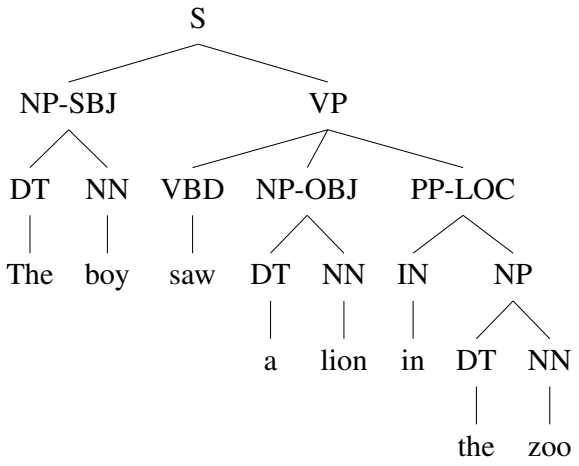


FIGURE 2.3: A phrase structure parse tree by using the annotation of the PTB.

The parse tree in Figure 2.3 shows the phrase structure annotation. It has a nominal subject 'The boy' which has been annotated by using a noun phrase (NP). A functional label 'SBJ' is further attached to represent the subject argument of the sentence. The second constituent annotates a verb phrase (VP) which has the main verb, an NP and a prepositional phrase (PP). The NP following the verb, represents the object of the sentence. English has a fixed subject-verb-object (SVO) word order which can be used to identify the core arguments. The attachment of the PP phrase is important to convey the proper meaning of a sentence. It shows the location of the action by using a LOC function label. The PP phrase in the example is attached under the VP which depicts that the action 'saw' occurred in the zoo. If the PP is attached to the NP, the lion is in the zoo but the boy may be in front of a tv set.The PP attachment is important to convey the actual meaning of a sentence. It is important to note that the phrase structure annotation is helpful to understand the syntactic structure of the sentence as well as the semantics associated with it.

TABLE 2.6: The Penn Treebank phrase labels.

S#	Phrase Label	Description
1	S	Simple declarative clause
2	SBAR	Clause introduced by a subordinating conjunction
3	SBARQ	Direct question by a wh-word or wh-phrase
4	SINV	Inverted declarative sentence
5	SQ	Inverted yes/no question
6	ADJP	Adjective Phrase
7	ADVP	Adverb Phrase
8	CONJP	Conjunction Phrase
9	FRAG	Fragment
10	INTJ	Interjection
11	LST	List marker
12	NAC	Not A Constituent
13	NP	Noun Phrase
14	NX	Used within certain complex noun phrases
15	PP	Prepositional Phrase
16	PRN	Parenthetical
17	PRT	Particle
18	QP	Quantifier Phrase
19	RRC	Reduced Relative Clause
20	UCP	Unlike Coordinated Phrase
21	VP	Verb Phrase
22	WHADJP	Wh-adjective Phrase
23	WHADVP	Wh-adverb Phrase
24	WHNP	Wh-noun Phrase
25	WHPP	Wh-prepositional Phrase
26	X	Unknown, uncertain or unbracketable

The functional labels are helpful to depict the grammatical relations but they lack the semantic information of the predicates. The predicate argument structure does not annotate the senses of the predicates and semantic roles. The PTB was further updated to have semantic roles as described in [88]. The roles have been numbered from one to five in the form of Arg:0, Arg:1, ..., Arg:5. These semantic frames were

TABLE 2.7: The Penn Treebank functional labels.

Fun. Label	Description	Fun. Label	Description
ADV	Adverbial	DIR	Direction
NOM	Nominal	EXT	Extent
DTV	Dative	LOC	Locative
LGS	Logical subject	MNR	Manner
PRD	Predicate	PRP	Purpose or reason
PUT	Locative complement	TMP	Temporal
SBJ	Surface subject	CLR	Closely related
TPC	Topicalized	CLF	Cleft
VOC	Vocative	HLN	Headline
BNF	Benefactive	TTL	Title

defined for frequent verbs.

Due to the fixed word order of English, the parsing models are usually trained by removing the functional label. In [57], the PTB was trained by including functional labels and empty categories. The experiments carried out by using the Collins' parser [38]. The results are quite comparative when parsing without functional labels. It is evident that for language with flexible word order, the annotation mechanism have the potential to represent the syntactic structure.

A number of treebanks have been produced by using phrase structure annotation after the development of the PTB. We describe the prominent phrase structure treebanks in this section. The Penn Chinese Treebank (CTB) has also been developed by using the phrase structure annotation due to its fixed word order. The CTB uses a POS tag set containing 33 tags, 20 phrase labels and 25 functional labels [154]. The treebank was further updated to add the information of semantic roles by annotating the propositions [153]. The inter-annotator agreements have been calculated against gold standard and a 20% reference corpus selected randomly. During the bracketing phrase the average

annotator accuracy was 96.7% and the average consistency score was 93.8%.

A phrase structure treebank has been developed for Arabic viz The Penn Arabic Treebank [103, 102]. The teebank has been developed by using PTB style of annotation and guidelines. Despite the difference of grammatical structure and word order, the PTB guidelines were able to represent the syntax of the language. English language has a fixed SVO order and Arabic usually has VSO but SVO order is also available in the language. The guidelines have been updated to infer the Arabic argument structure such that; the subjects are attached under VP phrases, clausal coordinations are annotated at the matrix clause level and nominal objects are annotated as NP-OBJ for transitive verbs. The Arabic treebank also has three layers of annotations, which are; POS tagging, phrase structure and functional layer. At initial level, the phrase annotation was performed by using a parser described in [17]. The structures were further reviewed by the annotators and functional labels were added. The inter-annotator agreement for POS tags has been computed for 853 words and the overall agreement score was 85% and the pair was agreement was 92.2%.

Another phrase structure treebank has been developed for Korean viz The Penn Korean Treebank [75, 74]. The treebank initially contained 54 thousand tokens and five thousand sentences. The treebank was further extended to a second version by including additional text from the domain of news [76]. The annotation guidelines were updated for newly encountered syntactic constructions. However, the main annotation scheme and guidelines remained the same. It uses eleven clause and phrase labels and a set of six functional labels. The annotation has been performed in the form of brackets like the PTB. To ensure the accuracy of the annotation, POS tagging and phrase labeling

have been evaluated after manual revision of the whole corpus. The POS tags have been evaluated to check the wrong or impossible tags, ungrammatical sequence of tags against word phrases and wrong choice of tag in case of applicability of multiple tags. The bracketing was evaluated by extracting a context free grammar rules which were further analyzed based on their syntactic structure and frequency in the corpus. The erroneous constructions were corrected manually by searching the annotated sentences. For the inter-annotator agreements, ten percent random corpus was annotated by two annotators and the agreement scores were satisfactory.

The development of a French treebank has been described in [4]. The treebank contains one million words from a newspaper domain. The resource provides the annotation for POS tags, morphology, lemmas and constituency. It uses 14 phrase labels. The treebank was further enriched to have a layer of functional labels by using eight labels [3]. The morphology and POS tags were evaluated manually to achieve the annotation accuracy. The lemmas were added automatically without manual intervention. The bracket labeling was performed automatically by the parsers which were further reviewed by the annotators. The evaluation was also done automatically by checking 500 sentences selected randomly. For opening brackets the f-score was 94%. For closing bracket the f-score was 56.5%.

A phrase structure treebank for Spanish has been described in [114]. It is a small treebank and contains 1,500 sentences with 22,695 words. The POS tagging was done automatically by using a tagger and a chunker was used to identify the phrase boundaries for frequent phrase like NP, ADJP, VP, ADVP and PP. The treebank has twelve phrase labels. The phrase annotation was evaluated by using a phrase structure

rule generator to detect the implausible constructions. Another treebank has been built for Spanish which is referred as AnCora treebank [145]. It contains two repositories, one for Catalan and the other for Spanish. It overall consists of 0.5 million words from each language. It was developed from two already existed resources Cast3LB[34] and a lexical resource for semantic annotation [9]. The annotation have different levels including; POS tags, Lemmas, phrase structure including functional labels and semantic structure. The semantic layer annotates argument structures, thematic roles, semantic verb classes, named entities and nominal senses from WordNet. These layers were annotated independent of each other. Spanish and Catalan both have flexible word order. The most frequent word orders in the corpus are SVO, SV, SOV and OSV which have occurrences of 28%, 27%, 2% and 2% respectively. Another Spanish treebank has been built as a part of SMULTRON (Stockholm MULtilingual TReebank) project [63] which initially contained English-German-Swedish parallel treebanks. It has been annotated by using 20 phrase labels. The POS and phrase labels have been derived from AnCora treebank. The current version of the SMULTRON (4.0) contains 4,075 sentences of Spanish with 90,229 tokens.

A number of phrase structure treebank have been developed for many languages including Italian[113], German[139], Portuguese[6, 58], Japanese[84], Swedish[119], Vietnamese[115, 116], Thai[127] and Hebrew[137]. All these treebanks have their own annotation label sets with respect to their constituency structure. Additional semantic layers are also added to mark grammatical functions, argument structures, semantic roles etc. We now discuss the prominent treebanks for Hindi and Urdu and their annotation styles in the following lines.

The phrase structure treebanks have been developed for many Western language. Most of them restrict the word orders and the order of arguments resulting a fixed hierarchical phrasal structure. However efforts have been done for the syntactic analysis of the South Asian languages. The Hindi-Urdu Treebank (HUTB) project [14] produces two annotated repositories for Hindi and Urdu. The HUTB treebanks have been developed by using dependency structure. We describe the dependency structure treebanking in Section 2.2.2. The phrase structure guidelines for both languages have been proposed in [15]. The proposed phrase structure has been inferred from the Minimalist Program and presents the binary trees by using specific positions of the arguments in a sentence. The position and the phrase structure of a typical transitive sentences is shown in Figure 2.4.

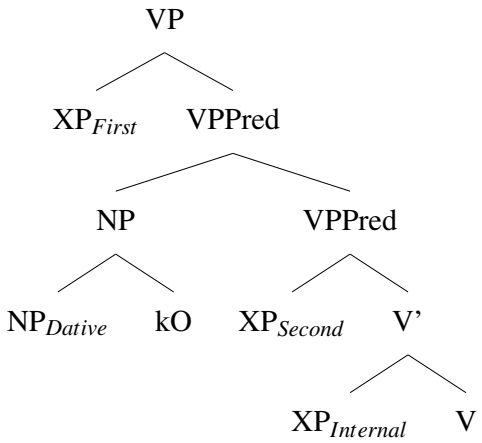


FIGURE 2.4: A proposed phrase structure annotation and argument positions from [15].

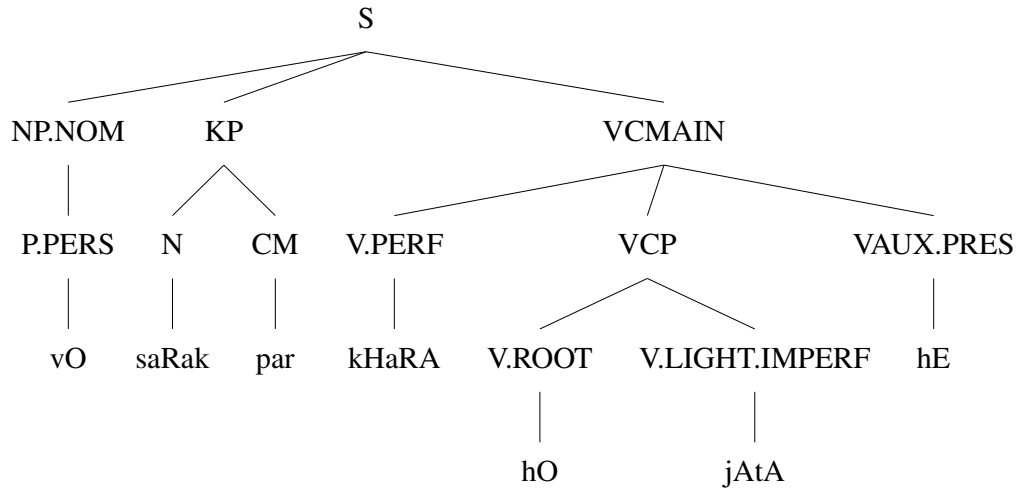
The first position of the tree in Figure 2.4 has been specified for ergative subjects which can show agreement with the verbs. Similarly, the dative position has been reserved for dative subjects which is identified by the 'kO' case marker. If a sentence does not have dative construction then this position is not annotated. The second position in the tree has been specified for object of the transitive sentences which further agree

with the verb. The forth position shows the internal arguments of the sentences which usually contain small clauses. For flexible argument orders, the positional scrambling mechanism is adopted which further complicates the annotation. The phrase structure annotation is not suitable for a language having flexible positions of the arguments like Urdu. It also complicates the analysis of the phrase structure with the dependency structure.

A phrase structure treebank has been developed for Urdu viz Urdu.KON-TB [1, 2]. The treebank uses multiple annotation layers including POS tagging, phrase labels and functional labels. The phrase labels further attach the morphological information with labels. It concatenates the information of tense, case, aspect and modality with labels. The treebank has been annotated by using a large annotation scheme which contains 26 phrase labels. However, the treebank contains only 1,400 sentences with an average of 13.73 tokens per sentence. The corpus has been mainly collected from Urdu Wikipedia[4] and the Jang newspaper[5]. Figure 2.5 shows a sample annotated sentence from the Urdu.KON-TB.

The parse tree in Figure 2.5 shows the attachment of three clauses under the 'S' phrase label. The 'S' label represents the annotation of the whole sentence which has an NP.NOM, a KP and a VCMAIN. The NP.NOM defines the nominal case along with a noun phrase. The POS tag for personal pronoun is P.PERS which marks the pronoun by depicting its type as well. The nominal case represents the nominal subject in this example. The second clause annotates a case marking clause with the label KP. The annotation of the case phrase is quite flat as noun and case marker appear at the same

[4]https://ur.wikipedia.org
[5]https://jang.com.pk

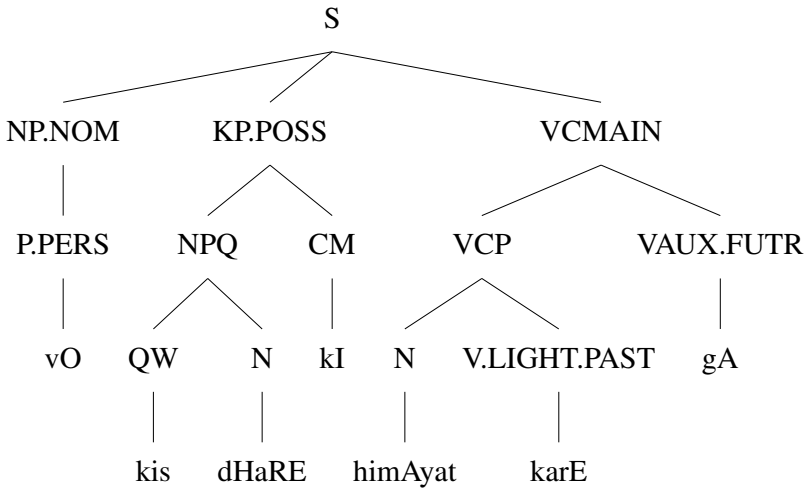


vO saRak=par kHaRa hO jAtA hE
Pron.3.M.Sg road.F.Sg=Loc stand.M.Sg be.Imper.Sg go.Pres.M.Sg be.Pres.3.Sg
'He stands on the road'

FIGURE 2.5: Parse tree from the Urdu.KON-TB [2].

level of the clause. The third clause defines the verbal structure with the VCMAIN label. The verbal structure contains main verb annotated with V.PERF which defines the verb and its tense, a VCP clause containing verb root and a light verb and finally an auxiliary verb by using a VAUX label along with tense information. The verbal construction is quite complicated as it shows main verb *kHaRA* 'stand.M.Sg' with a root *hO* 'be.Imper.Sg' and a light verb *jAtA* 'go.Pres.M.Sg'. The verbal construction actually is making a complex predication to denote adjective+verb construction. If the *kHaRa* is a verb in past form then what is its present form? This annotation also lacks to annotate the changed argument order of the complex predicate. The nouns or adjective do not always appear with nouns therefore a different annotation is necessary which has the ability to cater the flexible order of the language. The Urdu.KON-TB divides the verbal structure into four categories which include, VCMAIN; which annotates the main clause, VCP; to annotate complex predicates, VIP; which marks infinitive verb

phrase and VP for simple verb phrases. Figure 2.6 shows another annotated example.



vo	kis	dHaRE=kI	h2amAyat	karE-gA
Pron.3.M.Sg	which	group.M.Sg=Gen	support.F.Sg.Nom	do.Fut.M.Sg

'Which group will he support.'

FIGURE 2.6: Parse tree from the Urdu.KON-TB [2].

The annotation of the sentence shown in Figure 2.6 has a nominative subject, genitive case marking clause and a main verbal structure. The case marking clause shows the possessive case which is annotated to have a question noun phrase and a genitive case marker *kI*. The clause labels the possessor but word *himAyat* 'support' is attached with the VCP. In case of different word order, the annotation specifies the position of noun+verb complex predicate.

The treebank has a large annotation label set which contains the duplicate labels for questions like NPQ, KPQ, QWP, ADJPQ, ADVPQ, SBARQ and SQ. The position of question words in Urdu is also flexible [27]. The Urdu.KON-TB is a small treebank with a complex annotation scheme. The linguistically rich phrase label sets provide the deep linguistic insight of the language. To perform the statistical parsing, a relatively large annotated corpus is required with respect to annotation scheme. The language

specific annotation tag sets make the resources less usable. Therefore, the annotation should be compatible with existing resources.

A universal phrase label set has been propose for multiple languages in [73]. They devised a common and simplified label set for 25 treebanks from 21 languages. The proposed tag set was helpful to achieve better parsing results. Related phrase labels were combined into single labels. For example, additional phrase labels are used for wh-words and wh-clauses in the annotation of the PTB [16] which include; SQ, SBARQ, WHNP, WHPP, WHADJP and WHADVP. The universal tag set combines all types on noun phrases into a single NP and 'S' is used to annotate all types of clauses. Table 2.8 presents the proposed universal label set and its mapping from the PTB-I[107] and the PTB-II[16].

TABLE 2.8: The Mapping of the PTB-I[107] and the PTB-II[16] on universal phrase label set proposed in [73].

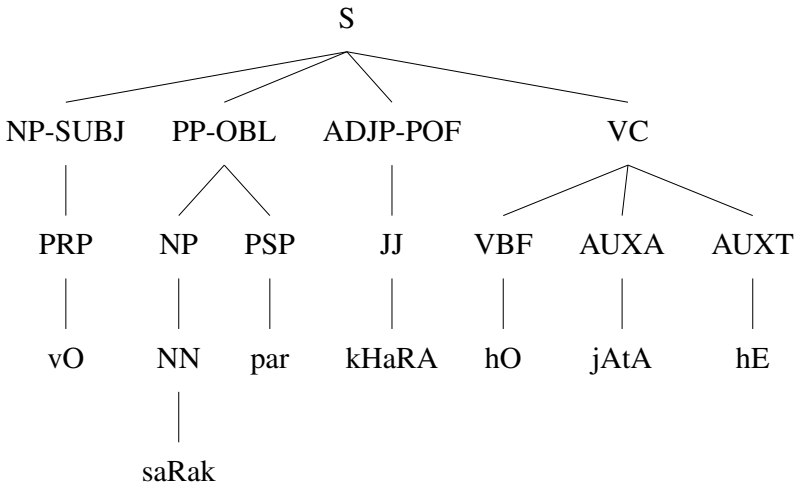
S#	Universal Label	English PTB-I	English PTB-II
1	NP	NP, WHNP	NP, NAC, NX, QP, WHNP
2	VP	VP	VP
3	AJP	ADJP	ADJP, WHADJP
4	AVP	ADVP, WHADVP	ADVP, WHAVP, PRT, WHADVP
5	PP	PP, WHPP	PP, WHPP
6	S	S, SBAR, SBARQ, SINV, SQ	S, SBAR, SBARQ, SINV, SQ, PRN, FRAG, RRC
7	CONJP	–	CONJP
8	COP	–	–
9	X	X	X, INTJ, LST, UCP

Table 2.8 presents the mapping of two versions of PTB on the universal tags. However the mapping has been done for 25 treebanks from 21 different language. The

proposed universal tag set has nine phrase tags. It has labels for noun phrase (NP), verb phrase (VP), adjective phrase (AJP), adverbial phrase (AVP), prepositional phrase (PP), sentence or clause phrase (S), conjunction phrase (CONJP), coordinate phrase (COP) and all other phrase labels are mapped on 'X' label. The mapping loses the deep linguistic structure of the parse trees and it would not guarantee that the mapping could be reversed back accurately. Although the simplified annotation offer certain benefits in parsing results. The universal labels can also be used as a common annotation scheme for other treebanks which reduce the annotation time and effort.

A phrase structure Urdu treebank (CLE-UTB) has been developed by using a version of the universal tag set [55]. The treebank uses the Urdu POS tag set which is proposed in [86]. The treebank further has a layer of functional tags on the phrase labels. The label sets have been inspired from the PTB [107, 16], Urdu.KON-TB [2], HUTB [14] and the universal tag set [73]. It makes the annotation scheme compatible with existing resources. Figure 2.7 presents the annotation of the CLE-UTB for the sentence from Figure 2.5.

The CLE-UTB attaches the arguments at the main clause level (S) which provides an ability to annotate any order of arguments in a sentence. The complex predicate structure is also represented without specifying its position in the sentence. The functional labels represent the grammatical relations like subject and oblique in the given example. A label POF annotated the complex predicate independent of its position on the sentence. The details of the phrase label set and annotation guidelines for the development of the CLE-UTB are presented in Chapter 3. The compatibility of the phrase annotation with dependency structure is discussed in Section 3.2.2 and Appendix A in



vO saRak=par kHaRa hO jAtA hE
Pron.3.M.Sg road.F.Sg=Loc stand.M.Sg be.Imper.Sg go.Pres.M.Sg be.Pres.3.Sg
'He stands on the road'

FIGURE 2.7: Annotation of the parse tree by using the annotation of [2].

more detail.

2.2.2 Dependency Structure Treebanks

The dependency structure (DS) is another grammatical framework which defines the relations of every word in a sentence. These relations are referred as dependencies which are further marked with labels to annotate core and non-core dependencies. In most cases, the main finite verb is taken as the center or root of a whole sentence and core arguments show their dependencies on the root. Other words derive non-core dependencies on their respective heads. The DS is flatter in nature as it shows relations among lexical components rather than bounding them in a hierarchical structure, in contrast with the phrase structure. The DS is considered more suitable for free word order languages but modern syntactic analysis are being performed by using dependency relations irrespective of the word order of a language. Figure 2.8 presents the dependency

representation of the phrase structure parse tree from Figure 2.3.

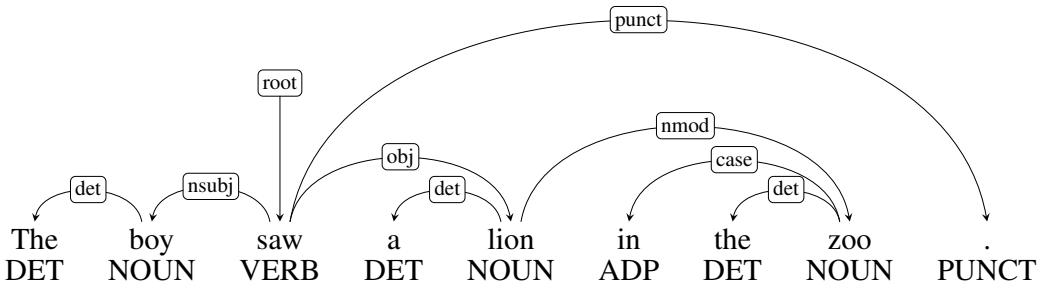


FIGURE 2.8: An equivalent dependency structure tree from Figure 2.3 by using universal dependency labels.

The dependency tree in Figure 2.8 shows two core arguments by using labels `nsubj` and `obj` on the dependency arcs. The word 'boy' is the subject and the word 'lion' is the object. Both arguments show a dependency on the root which is the main verb of the sentence. Beside the core arguments, all other tokens have been represented with an arc and a label. For example, the tokens 'the' and 'a' are determiners of their respective heads and are defined by `det` dependency label. The preposition is defined by a `case` label and the head word 'zoo' has the dependency on the head of the object which has a noun modifying relation. It is defining the location of the lion. The punctuation also has a dependency relation with the root. The annotation of the sentence has been performed by using Universal Dependencies Version-2 (UD-2)[6] [121, 118, 122]. Universal Dependencies (UD) provides a language independent framework to annotate any language by using dependency structure. The framework has the ability to perform syntactic structure, POS and morpho-syntactic features. Table 2.9 shows the latest UD labels for POS tagging and syntactic annotation by using the release V2.5 [155].

[6]https://universaldependencies.org/

TABLE 2.9: Universal Dependency UD-V2 labels.

Categories	Nominals	Clauses	Modifier words	Function words
Core arguments	nsubj	csubj		
	obj	ccomp		
	iobj	xcomp		
Non-core dependents	obl	advcl	advmod	aux
	vocative		discourse	cop
	expl			mark
	dislocated			
Nominal dependents	nmod	acl	amod	det
	appos			clf
	nummod			case
Coordination	MWE	Loose	Special	Other
conj	fixed	list	orphan	punct
cc	flat	parataxis	goeswith	root
	compound		reparandum	dep

The universal dependencies have been derived from Stanford Universal Dependencies [45, 44, 46]. Currently there are 157 dependency treebanks for 90 different languages which have been annotated by using UD-V2.5 dependencies [155, 122]. Table 2.9 presents 37 dependency labels for different syntactic categories including core and non-core arguments, nominal dependents etc. The universal dependencies also provide the cross language annotation guidelines. The core arguments include the labels to annotate nominal subjects, direct and indirect objects, clausal subjects, clausal complements and open class complements. The non-core dependencies mark obliques, vocatives, expletive, dislocated elements, adverbial clause modifiers, adverbial modifiers,

discourse elements, auxiliaries, copula and mark. The third category include the dependencies for nominal modifiers, clause modifiers for nouns, apposition modifiers, numeric modifiers, adjectival modifier, determinants, classifiers and case. There are other dependencies for coordinations, multi-word expression, loose and some miscellaneous dependencies. The universal dependencies also provide a cross language framework to mark POS tags. Table 2.10 shows the universal POS tags.

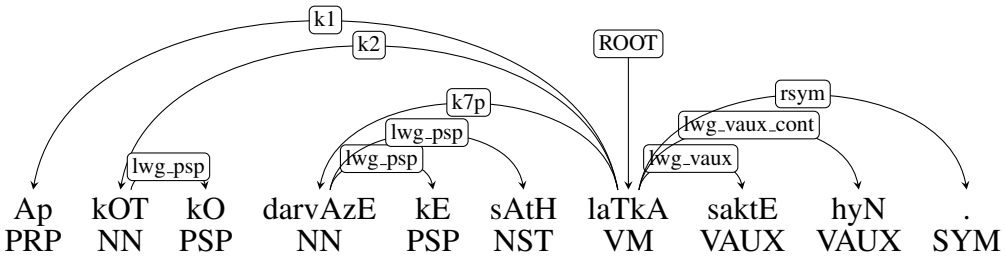
TABLE 2.10: Universal Dependency UD-V2 labels.

Open class words	Closed class words	Other
ADJ	ADP	PUNCT
ADV	AUX	SYM
INTJ	CCONJ	X
NOUN	DET	
PROPN	NUM	
VERB	PART	
	PRON	
	SCONJ	

There are six universal POS tag to mark open class words and eight tags to mark closed class words and three tags to mark punctuation and symbols. The unspecified word classes are tagged by using 'X' tag.

Some treebanks have been converted to UD labeling automatically, we especially discuss, the Hindi-Urdu Treebank (HUTB) [14, 144]. The HUTB was originally annotated by using a Paninian grammatical framework [11]. The HUTB also performs the morphological analysis by including the information of lexical categories, number, gender, person, case, TAM (tense, aspect and mood) along with POS. The model provides semantic analysis along with syntactic annotation of a sentence. The information

is presented in two layers, at first layer it produces the semantics of words and at second layer it defines the role of the words that they play in the relations with other lexical components. Figure 2.9 presents the annotation of an Urdu/Hindi sentence by using the Paninian grammatical framework used in [14].



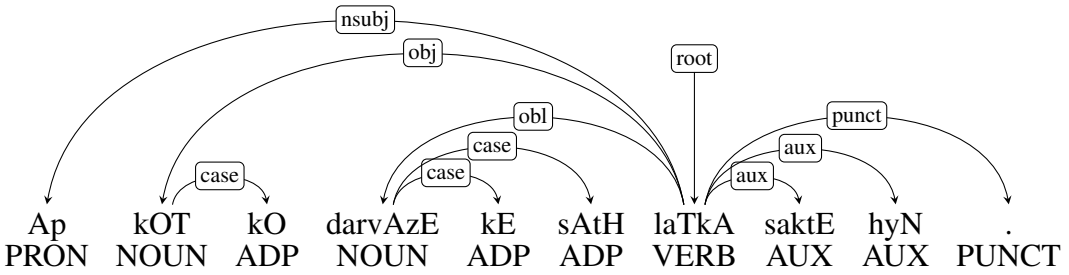
Ap kOT=kO darvazE=ke sAtH laTkA saktE hyN
Pron.2.M.Pl coat.M.Sg.Acc door.M.Sg.Gen with hang.Impr.M.Sg can.M.Pl be.Pres.3.Pl
'You can hang the coat on the door.'

FIGURE 2.9: The dependency annotation of an Urdu/Hindi sentence by using Paninian grammatical model[11].

The Paninian framework proposed six main participants of a sentence which are called *Karakas* 'participants'. However a sentence may have several other relations like purpose, reason, genitives etc. The HUTB use additional labels to annotate other dependency relations along with basic participants. The labels are constructed using letter 'k' which is followed by a number to represent specific *Karaka* labels. The HUTB has been released in two formats, the CoNLL format and Shakti-Standard Format (SSF) [12]. The HUTB annotates the agent or subject by using the label *k1* 'kartaa' and the object by using a label *k2* 'karma' as shown in Figure 2.9. The location in space has been marked by the label *k7p* 'deshadhikarana'. The finite verb is annotated as root of the sentence. Several other labels are used to mark dependency relations among the words. For example, the label *lwg_psp* is used to mark the genitive case and two labels

lwg_vaux and *lwg_vaux_cont* have been used to mark auxiliary verbs. A comparison of HUTB label set is shown in Table B.1.

The HUTB has been converted to universal dependency as described in [144]. Figure 2.10 shows the representation of the dependency structure from Figure 2.9 by using universal dependencies.



Ap kOT=kO darvazE=ke sAtH laTkA saktE hyN
Pron.2.M.Pl coat.M.Sg=Acc door.M.Sg=Gen with hang.Impr.M.Sg can.M.Pl be.Pres.3.Pl
'You can hang the coat with the door.'

FIGURE 2.10: The universal dependency annotation of the sentences from Figure 2.9.

The CLE-UTB [55] is a phrase structure treebank, however its annotation is compatible with dependency structure [54]. Section 3.2.2 and Appendix A present the automatic conversion of the CLE-UTB to universal dependencies and dependency parsing.

2.3 STATISTICAL PARSING

A statistical parser provides the analysis of a sentence and assigns the syntactic categories. The syntactic structure is helpful to understand the semantics. The syntax is represented in the form of a parse tree. A parse tree shows the break down of a sentence into sub-phrases. A parse tree contains a root, non-leaf nodes and leaf nodes. The POS tags are referred as pre-leaf nodes as the parsers predict the phrase level relations.

However, the accuracy of POS tags is influential on the parsing scores. The syntactic annotation of a language faces an issue of structural ambiguity. Generally, the ambiguity is referred as the possibility of multiple parse trees hence with multiple meanings of a single sentence. The structural ambiguity is quite common in the natural languages. We describe different types of parsers in the following sections.

2.3.1 Grammar-based Parsers

A probabilistic context-free grammar (PCFG) is derived from an annotated corpus and is used for the prediction of parse trees for unseen sentences. The grammar rules are treated independent without using the contextual information. Each rule has a probability in the grammar. The probability is calculated by dividing the frequency of a rule by the frequency of the rule head. The left-hand side of a rule is referred as head of a rule. The head frequency is computed by counting the rules containing same head throughout the grammar of a treebank [81]. Equation 2.1 shows the process to compute independent probabilities for grammar rules.

$$P(A \rightarrow B) = \frac{Count(A \rightarrow B)}{\sum_R Count(A \rightarrow R)} = \frac{Count(A \rightarrow B)}{Count(A)} \tag{2.1}$$

B and R are sequences of terminals and non-terminal symbols whereas the A is a non-terminal. A is the head of the rule $A \rightarrow B$. The sum $\sum_R Count(A \rightarrow R)$ presents the count of rules in which the head is A. The probability of a parse tree is computed by the product of probabilities of rules in the parse tree. The probability of a parse tree $P(T)$ can be calculated as shown by Equation 2.2.

$$P(T,S) = P(T) = \prod_{i}^{x} P(A \rightarrow B) \tag{2.2}$$

T is the parse tree of sentence S. A is the head of a production rule $A \rightarrow B$ and B is the rule body. i is the number of first rule of total x rules. The most probable parse tree can be computed by maximizing tree probabilities as depicted by Equation 2.3. The highest probability $P(T)$ parse tree is considered as predicted tree against the sentence S.

$$P(S) = \arg \max_{T \in parses(S)} P(T) \tag{2.3}$$

PCFGs make efficient parsers with respect to processing speed as they do not compute contextual information when implemented straight away. Context of phrases in a parse tree could be helpful to achieve more accurate parses. A few treebank representations could be helpful to embed contextual information with PCFGs. Section 4.3.2 describes the details of treebank representations for parsing with PCFG based parsers.

Lexical conditioning can provide the contextual information to the parse trees. The lexicalized grammars use the lexical information from phrases and sub-phrases. For this purpose, phrasal heads are computed to acquire the syntactic contribution of dominant words within the phrases. The head based parsing was proposed in [104] which was further used to achieve better parsing results by [38, 31, 94]. The head word computation is a language dependent task. The lexical items have been used to extract the treebank grammar in [104]. The parsing has been performed by using decision trees in left to right and bottom-up fashion. The parse trees were annotated with an array of

features. The features included nonterminal symbols, lexical head word, POS tag and extension of the siblings in the tree. The extension told that whether the next sibling is on the left or right-hand side of the child or is it going up to the parent node to become the head of the constituent. Words and their POS tags were selected to become the head of the nodes. The parser achieved precision and recall of 86% for sentences upto length of 40 words.

Another statistical parsing technique was introduced in [40] which used the lexical information along with bigram word dependencies. The head model was used from [104]. In addition, head word dependencies were used to compute the probabilities for parse trees in parsing. The co-occurrences of words and POS tags were used to deal the data sparsity. The backing-off method was used to estimate the dependent probabilities. The parser produced compatible parsing results with precision of 86.3% and recall of 85.8% for sentences having length upto 40 words on Wall Street Journal's standard train and test division of the Penn Treebank.

Furthermore, well-known three head-driven statistical parsing models were developed by Collins [36, 38]. The models compute lexicalized probabilistic context-free grammars for estimation. The head finding rules are presented in Table 2.11 from [37]. The heads are computed on the basis of the head finding directions and tag priority list for each phrase label. Due to data sparsity, some independent assumption were performed in Model-1. It considers the probabilities of right-hand side of a grammar rule to compute the probability of left-hand side. It further assumed the modifiers as independent of each other. Model-1 also used the distance factor in the conditional probabilities from [40]. The Model-2 was based on the Model-1 and it further annotated the grammar

with the complements, adjuncts and information of sub-categorization. It helped in the identification of subjects, objects and adjuncts in the sentences. The verbal head was also helpful to identify the number of arguments a verb takes. The Model-3 provided the probabilities of wh-movement and traces. Wh-movement was handled by adding a gap feature to the non-terminals of a parse tree. The parsing results from these three models were further improved in [38] as compared to [36]. The parsing results presented in [38] are; labeled recall and precision of 87.9% and 88.2% from Model-1, labeled recall and precision of 88.5% and 88.7% from Model-2 and the Model-3 performed with labeled recall of 88.6% and labeled precision of 88.7%.

Another head word based parsing techniques was proposed in [31]. The conditional probabilities have been used for head words and grammar rules for estimations. The grammar rules were not computed independently. The parser computed the probabilities for a head by giving its type, head word of the parent and the type of the parent node. These features caused data sparsity, therefore the deleting interpolation method was used for smoothing. Furthermore, the probabilities of the constituents was computed by its context. The parser performed with a labeled recall of 87.5% and a labeled precision of 87.4% for sentences of length up to 40 words.

A probabilistic feature grammar (PFG) was introduced in [65]. Morphological aspects of the head words were exploited for parsing. They used the contextual information of the grammar rules from parent labels. A grammar rule contained the node types, head words and their morphological information. Independence assumptions were taken to overcome the data sparsity. For using POS tags only, the parser performed with a labeled recall of 81.0% and labeled precision 82.2% for sentences

TABLE 2.11: The Head rules for The Penn Treebank from [37].

Phrase label	Direction	Priority List
ADJP	Left	NNS QP NN $ ADVP JJ VBN VBG ADJP JJR
		NP JJS DT FW RBR RBS SBAR RB
ADVP	Right	RB RBR RBS FW ADVP TO CD JJR JJ
		IN NP JJS NN
CONJP	Right	CC RB IN
FRAG	Right	
INTJ	Left	
LST	Right	LS :
NAC	Left	NN NNS NNP NNPS NP NAC EX $ CD QP
		PRP VBG JJ JJS JJR ADJP FW
PP	Right	IN TO VBG VBN RP FW
PRN	Left	
PRT	Right	RP
QP	Left	$ IN NNS NN JJ RB DT CD NCD
		QP JJR JJS
RRC	Right	VP NP ADVP ADJP PP
S	Left	TO IN VP S SBAR ADJP UCP NP
SBAR	Left	WHNP WHPP WHADVP WHADJP IN DT
		S SQ SINV SBAR FRAG
SBARQ	Left	SQ S SINV SBARQ FRAG
SINV	Left	VBZ VBD VBP VB MD VP S SINV ADJP NP
SQ	Left	VBZ VBD VBP VB MD VP SQ
UCP	Right	
VP	Left	TO VBD VBN MD VBZ VB VBG VBP VP
		ADJP NN NNS NP
WHADJP	Left	CC WRB JJ ADJP
WHADVP	Right	CC WRB
WHNP	Left	WDT WP WP$ WHADJP WHPP WHNP
WHPP	Right	IN TO FW

having length up to 40 words. On the basis of head words, the parsing results were improved to labeled recall of 84.8% and labeled precision of 85.3%.

Another lexicalized parser has been developed in [94] along with dependency parser by using A* algorithm. They updated the head rules for lexicalized probabilistic context-free grammar. They added a rule for NX and updated several other rules from [37]. The lexicalized model performed with a labeled recall of 86.8% and a labeled precision of 86.6% with an F-score of 86.7%.

The lexicalized grammars produce lexical contextual information during the probability estimation of the parse trees hence with improved parsing results. However, the lexical conditioning causes an issue of data sparsity requiring large amount of annotated data for training the parsers. The contextual information can also be encoded without lexical conditioning. An annotation technique was proposed in [80]. The inner nodes are annotated with the name of the parent node. The annotation excluded the root and the pre-terminal nodes. The proposed annotations were not programed to perform the parsing but they presented the results of the tree transformations.

The treebank representation have been used to perform constituency parsing. In [93], an unlexicalized PCFG has been implemented for parsing. The model uses the parent annotation to include the contextual information. Pre-terminals were also annotated for the parse trees. The parsing results were quite comparable with lexicalized PCFG with a labeled recall of 85.7% and a labeled precision of 86.9% for sentences having length up to 40 words.

Another unlexicalized grammar-based parser has been proposed in [125]. The parser splits and merges the non-terminals on the basis of the highest likelihood. As

compared to [80] and [93], the grammar is more accurate and compact. The inside-outside probabilities were computed to develop a split and merge algorithm. The parsing results were quite promising as compared to lexicalized parsers. The parser performed with a labeled recall of 90.0% and a labeled precision of 90.3% for sentences with length of 40 or less. The parser also performed well on longer sentences and produced a labeled recall of 89.6% and a labeled precision of 89.8 for sentences of length up to 100 words. The parsing results were further improved in [124] by using a coarse to fine method for incremental pruning from hierarchical projections of the grammar and different inference methods for splitting PCFGs. The parser improved the results by a labeled recall of 90.5% and labeled precision of 90.7% for the sentences of length less or equal to 40 and a labeled recall of 89.9% and a labeled precision of 90.2% for sentences of length up to 100. We have discussed more prominent grammar based parsers in this section. Next section describes another formalism which incorporates the contextual information by learning sub-trees rather just grammar rules.

2.3.2 Data-oriented Parsing

Tree Substitution Grammars (TSG) make probabilistic parsers. They were proposed by [131] and were further formalized by [20]. The parsing based on TSGs is also called data-oriented parsing (DOP). DOP model carters all possible subtrees in a parse tree and predicts parses for unseen sentences. These subtrees are called fragments. Several variations of DOP model are available with different applications. The DOP model can produce a huge number of fragments with respect to number of parse trees in a treebank. Many fragments may produce a lot of candidate parse trees for one sentence.

Figure 2.12 presents all possible fragments [157] for a sentence from Figure 2.11.

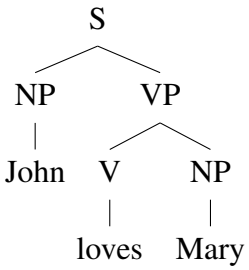


FIGURE 2.11: A sample parse tree.

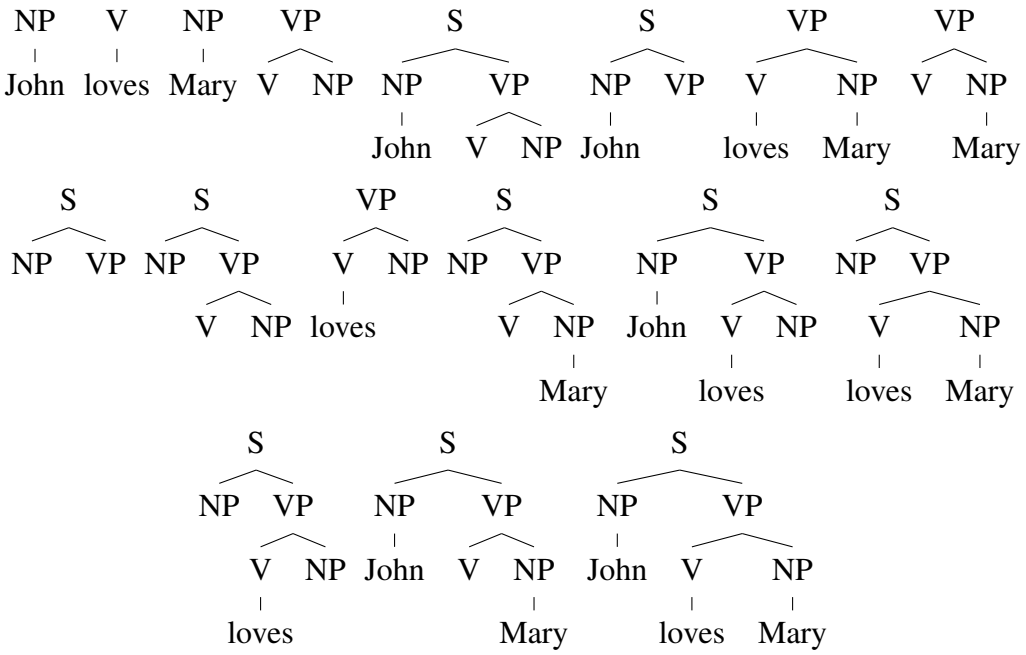


FIGURE 2.12: Fragments of the parse tree from Figure 2.11.

The DOP model computes the probability distribution to perform disambiguation. It calculates fragment probabilities and derivations for estimation. The sum of probabilities of all fragments is equal to one as shown by Equation 2.4.

$$\sum_{f \in Fx} P(f) = 1 \tag{2.4}$$

In Equation 2.4, Fx represents the set of all subtrees with root x. A sequence of fragments which produces a parse tree t by using the left-most substitution is called a derivation. Equation 2.5 shows the probability calculation for a derivation d.

$$P(d) = \prod_{f \in d} P(f) \tag{2.5}$$

The relative frequency estimate (RFE) produces the simplest probability estimate against a fragment . RFE is calculated by dividing the frequency of a fragment f with sum of all fragments having same root in the treebank.

$$P_{RFE}(f) = \frac{Count(f)}{\sum_{f' \in Froot(f)} Count(f')} \tag{2.6}$$

RFE does not provide a complete probabilistic interpretation. It contributes a biased estimate by assigning a higher probability value to larger fragments. Therefore, [66] proposed an equal weight estimate as shown by Equation 2.8.

$$W_{EWE}(f) = \sum_{t \in TB} \frac{Count(f,t)}{|\{f' \in t\}|} \tag{2.7}$$

$$P_{EWE}(f) = \frac{W_{EWE}(f)}{\sum_{f' \in Froot(t)} W_{EWE}(f')} \tag{2.8}$$

Equation 2.7 adds the division of sums against a fragment f for a tree t belonging to a treebank (TB) with all fragments. *P(f)* can be calculated by dividing the equal weight estimate by the count of all equal weight estimates of all fragments having same

root. To calculate the probability of a tree t, DOP counts the probabilities of all derivations of a tree t and maximizes the likelihood for the estimation of most probable tree as presented by Equations 2.9 and 2.10.

$$P(t) = \sum_{d \in D(t)} P(d) = \sum_{d \in D(t)} \prod_{f \in d} P(f) \tag{2.9}$$

$$t = argmax P(t) \tag{2.10}$$

The DOP model produced the contextual information by considering all possible subtrees. We have performed experiments for different subtree heights. Higher tree depths are helpful to attain more context of constituents.

2.3.3 Neural Parsers

In the previous few years, the statistical methods have been more focused to different types of neural networks. Neural networks started pushing state of the arts in the field of artificial intelligence, machine learning and natural language processing. In this section, we discuss neural models for constituency parsing.

A compositional vector grammar (CVG) was introduced by [141] which performs syntactic parsing by learning syntax and semantics of words in the treebank. They combined the syntactic categories achieved from a grammar based parser with a recursive neural network (RNN) model. The technique improved parsing results for English and achieved an f-score of 90.4. The parser was a continuation of parsing model proposed in [140]. Word embeddings were trained along with the parsing which were

proposed in [149]. The POS tags were appended with the word vectors so that each pair contains a word representation in the form of vector and POS tag. The CVG computes a score by adding up scores for all nodes of a tree. The parser is trained with max-margin objective function and predicts the tree with the highest score. The objective function and optimization used for training are described by Equations 2.11, 2.12 and 2.13.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} r_i(\theta) + \frac{\lambda}{2}||\theta||_2^2 \tag{2.11}$$

$$r_i(\theta) = max_{\widehat{y}\in Y(x_i)}(s(CVG(x_i,\widehat{y})) + \Delta(y_i,\widehat{y})) - s(CVG(x_i,y_i)) \tag{2.12}$$

$$\Delta(y_i,\widehat{y}) = \sum_{d\in N(\widehat{y})} k\{d \notin N(y_i)\} \tag{2.13}$$

Where, input sentence is represented as x_i, all possible parse trees for x_i are represented by $Y(x_i)$, correct prediction for x_i is y_i, $\Delta(y_i,\widehat{y})$ is the margin loss, k was set to 0.1 for all experiments, (x_i,y_i) represents all training samples, score function is represented by s, parameter collection is represented by θ, $\lambda = 10^{-4}$ for training and $N(y_i)$ represents all nodes for the parse tree y_i.

The word vectors have been used to find the correct parent for a pair of child nodes ($P \rightarrow AB$). A parent and its children have the same dimensionality. The vectors from child nodes are concatenated which result to have a dimension of $2n \times 1$. The

dimension of weight matrix was set to $x \times 2n$ and $tanh$ was used for nonlinearity. Similarly, the output vector was used for the prediction of parent nodes. CVG considers the syntactic constructions extracted from the treebank by using a PCFG parser. The weights were learned by training a syntactically-united recursive neural network (SU-RNN). Child nodes were also influential in the weight matrix. A node score was computed by adding up the linear score of a parent and log probability of the rule ($P \rightarrow AB$) as demonstrated by Equation 2.14.

$$s(p) = \left(v^{(A,B)}\right)^T p + \log P(P \rightarrow AB) \tag{2.14}$$

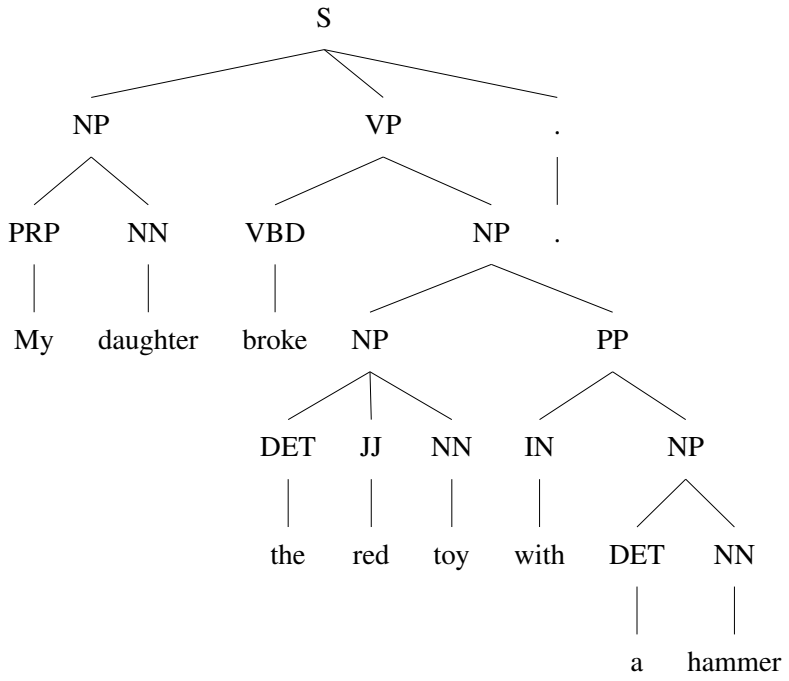
Where v is the parameter vector and the CVG computed scores for a parse tree by summing all node scores.

$$s(CVG(\theta, x, \hat{y})) = \sum_{d \in (\hat{y})} s\left(p^d\right) \tag{2.15}$$

A lexicalized PCFG parser has been used for caching top 200 parse trees against each training sample of the CLE-UTB. RNN parser has been trained with an objective as shown in Equation 2.11. Beam search was used to optimize the speed for top 200 parses.

Recurrent neural networks (RNNs) are able to provide a learning framework to model word sequences by using the entire context. LSTMs were introduced by [78] and were improved by many others [60, 61, 59, 68, 10, 69]. A constituency parser was developed by using LSTMs in [62]. The parser converts trees into linear structures along with tree labels and POS tags. The BiLSTM model was quite capable to predict

the sequential labels. The constituency parser described in [62] performed with best f-score of 90.6% by using word embedding, character embeddings and two hidden LSTM layers.



Absolute scale:
2NP 1S 2VP 4NP 4NP 3NP 4PP 5NP 1S
Relative scale:
2NP -1S 1VP 2NP 0NP -1NP 1PP 1NP -4S

FIGURE 2.13: A sample parse tree along with linearized labels.

Figure 2.13 shows a linear labeling from a constituent parse tree from [62]. The absolute labels have been achieved by counting common number of predecessors concatenated with first common phrase label from bottom to top. The relative labels have been achieved by subtracting the previous token's label number from the current token. The linear labels were used to train long-short term memory networks for prediction. The results were further converted back to tree structure for evaluation and the achieved f-score is 90.6%.

A chart-based neural model has been proposed in [143] which achieved improved parsing results with an f-score of 91.8% on PTB. They further used deep bidirectional long-short term memory (LSTM) neural networks to enhance sentence representations, designing sophisticated strategies for span representation. On the other hand, they adopted top-down incremental parsing for decoding, which simplified the differences between transition-based and chart-based approaches. Their results were quite competitive as compared with transition-based parsers. The technique was further followed by [91] by using the word representations which are contextualized word representations ELMo [123] and BERT [49]. In [91], self-attentive features were used for learning and the parsing f-score was 93.5% which is quite high for PTB. The model was further enhanced by using ELMo embedding. The ELMo embeddings produced deep contextualized word representations and the improved results were phenomenal with f-score of 95.1% on PTB.

On the other hand, a transition-based parsing system defines transition states and actions, where states denote partial parsing outputs, and actions specify next step state-transition operations. Several methods exist to implement transition strategies. The recurrent neural network grammars (RNNG) were proposed in [53] which is a top-down transition system. This model achieved a parsing f-score of 92.4%. In [99] an in-order transition system was developed which makes a compromise between top-down and bottom-up transitions. The model achieved an f-score of 91.8%. [92] suggested a system which used four types of tagging, it combined sequence tagging and transition action classification. They further used the BERT representation in the model training. They achieved a state-of-the-art parsing f-score of 95.4% on PTB dataset.

2.3.4 Parsing Evaluation

The constituency parsing results have been computed by using the Parseval measures [19]. The measure computes labeled precision, labeled recall and f-scores with respect to constituents as given below.

$$Labeled\ recall(LR) = \frac{\#\ Correct\ constituents\ in\ candidate}{\#\ Constituents\ in\ reference} \tag{2.16}$$

$$Labeled\ precision(LP) = \frac{\#\ Correct\ constituents\ in\ candidate}{\#\ Constituents\ in\ candidate} \tag{2.17}$$

$$Labeled\ F-score(F_1) = \frac{2 \cdot LP \cdot LR}{LP + LR} \tag{2.18}$$

For the constituency parsing, the candidate results are compared with the gold standard or reference corpus. For a constituent to be correct in the candidate set, its label and brackets both need to be correct. For example, an NP(0:2) defines a noun phrase contain a bracketing span from token '0' to '2'. Suppose the candidate parse tree for this specific example also contains an NP but with bracket span of '0:3'. In this case, the shown constituent is considered as incorrect. For a sentence of length 'n', the clause label S has a span from token number '0' to 'n-1'. The constituency parsing models, in this work, are evaluated by using Parseval measures.

Dependency parsers, on the other hand, are evaluated using different measures which include, labeled attachment score (LAS), unlabeled attachment score (UAS) and

label accuracy (LA). The dependency results are evaluated more like tagging measures. For LAS, each token is checked for its correct dependency label and correct syntactic head. If both are correct with respect to gold corpus, it is computed as correct with respect to all tokens in the candidate set. Similarly, the UAS just checks for the syntactic head attachment without labels. The LA computes the accuracy of dependency labels like POS tags.

2.4 PARSING MORPHOLOGICALLY-RICH LANGUAGES

All the statistical models discussed above are designed and developed to parse Penn Treebank for English. Adopting these models for other natural languages may or may not work properly. However, several adaptations have been tried for morphologically rich languages (MRLs). A head-driven parser has been implemented for Czech by using the Model-2 of Collins' parser [39]. Czech is a highly inflectional language and a relatively free word order language like Russian, Slovak, Slovene, Polish, Ukrainian and Serbo-Croatian. The model, after few alterations, has given moderate parsing scores although the parser has achieved more than ninety percent accuracy for English while parsing Wall Street Journal section of Penn Treebank. German is rich in morphology and also has free word order [147] but the parsing results are still low as compared to English [95].

An implementation of statistical parsing with lexicalized dependencies has been presented in [51]. They have concluded that un-lexicalized parser performs well as compared to lexicalized. They have shown that sister-head dependencies work better

than head-head dependencies for a flat structured tree bank. Further improvements have been done to German parsing by adding case and morphological information in [50].

A simple probabilistic context free grammar (PCFG) with morphological information and head annotation has outperformed the head-driven parsing models for Modern Hebrew [146]. Similarly, [41] has used the Collins' model-2 to parse Italian ISST treebank and the results are significantly low. Further adding the parent annotation and horizontal markovization has not done well either. Lexicalized parser with lemmatization improved the parsing results for French [134].

The concept of feature annotation has been introduced in [65]. Probabilistic Feature Grammar (PFG) has been presented to parse the Penn Treebank along with the lexical information. The formalism has not beaten the head-driven models for English but the idea has been used by the several researchers to parse the morphologically rich languages. [47] has used morphological features to discover the effect on parsing results. They have selected fifteen morphological features which are identified from the Penn Arabic Treebank [103] and have reported an improvement in parsing results. In the same way, morphological features have been used for Arabic dependency parsing [108]. Word clustering technique has also been used to decrease the effect of data sparsity while parsing French [135].

Parsing the morphologically rich language, like Urdu, is not a trivial task. To parse such languages, some specific issues need to be solved as discussed in [147, 148]. They have raised three questions which are needed to be answered while parsing an MRL. First question is regarding language representation and input type. Second question asks about the morphological information, whether it is encoded at part of speech

tags or at the non-terminal nodes. Third question is about the data requirements for training. Urdu is a morphologically rich language which has many inflectional forms for single word. In case of compounding where more words combine to provide sole meaning, how should it be considered, single word or multiple words? Part of speech tagging is a morpho-syntactic layer so which morphological features (POS tags, lemmas, number, gender, tense, honor) would be useful in parsing if at all? Data sparsity is also an issue for Urdu statistical parsing due to its nature to have many inflectional forms. This research is motivated to answer these questions by implementing the statistical constituency parser for Urdu.

3. URDU TREEBANK DEVELOPMENT

In this chapter, we present the process of treebank development and its evaluation. The chapter division is as follows. Section 3.1 discusses label sets which have been used to annotate the CLE-UTB, Section 3.2 presents the annotation guidelines by showing parse tree examples, Section 3.3 describes the corpus and its preparation for annotation, Section 5.1 discusses multi-step treebank evaluation process including inter-annotator agreement and treebank consistency checking and Section 3.5 shows the statistics of the treebank after development and evaluation.

3.1 LABEL SETS

Section 3.1.1 discusses the part of speech tag set, Section 3.1.2 describes the details of phrase labels and Section 3.1.3 presents the functional labels which have been used to mark grammatical roles.

3.1.1 POS Tag Set

The part of speech tag set for Urdu has been gone through several revisions. Hardie (2003)[77] proposed a POS tag set which contained 280 tags. To cover the morphology of the language, the tag set contained tags for a large number of surface forms which resulted in a lot number of tags. To train a statistical parser a sufficient

size of the data set and optimal number of POS tags are necessary. Too many tags will make it impractical to train a tagger due to lack of training samples in the corpus. Sajjad (2007)[77] and Sajjad and Schmid (2009) [129] performed automated part of speech tagging for Urdu by using 42 tags. Furthermore, the tag set was simplified to have 35 tags [86] and is referred as CLE Urdu POS tag set[1]. Table 3.1 shows the Urdu POS tag set proposed in [86] which has been used for the development of the CLE-UTB.

The tag set resembles the well-know Penn Treebank tag set. Due to morpho-syntactic properties of Urdu, several distinctive tags have been added. The tag set was used to train an Urdu corpus containing 100 thousand word. The Tree Tagger [132] was trained on the tagged corpus which produced an accuracy of 96.8% [86]. The tag set has a flat tagging mechanism for several categories. For example, it uses a single tag NN for all types of common nouns irrespective of their number and gender. Similarly, main verbs have been divided into two categories infinitive and finite verbs with tags VBI and VBF. However, there are four auxiliary verbs which are marked with four separate tags. Furthermore, cases are represented with case markers in Urdu and the tag set proposed a single tag for all kind of case markers. Urdu has a rare occurrences of prepositions and hence a single tag PRE to mark them. The punctuation symbols are also tagged with a single tag PU. The POS tag set defines the leaf nodes in parse trees. Therefore, a POS tag set should be able to mark all syntactic categories of any language. The CLE Urdu POS tag set provides an optimal number of tags which are also appropriate to tag Urdu language.

[1] http://www.cle.org.pk/software/langproc/POStagset.htm

TABLE 3.1: The CLE Urdu POS tag set.

Sr#	POS tag	Category
1	NN	Common noun
2	NNP	Proper noun
3	VBI	Infinitive verb
4	VBF	Finite verb
5	AUXA	Aspectual auxiliary
6	AUXP	Progressive auxiliary
7	AUXT	Tense auxiliary
8	AUXM	Modal auxiliary
9	PRP	Personal pronoun
10	PDM	Demonstrative pronoun
11	PRS	Possessive pronoun
12	PRD	Relative demonstrative pronoun
13	PRR	Relative personal pronoun
14	PRF	Reflexive pronoun
15	APNA	Reflexive apna
16	JJ	Adjective
17	Q	Quantifier
18	CD	Cardinal
19	OD	Ordinal
20	FR	Fraction
21	QM	Multiplicative
22	RB	Common adverb
23	NEG	Negation adverb
24	PRE	Preposition
25	PSP	Post-position
26	CC	Coordinate conjunction
27	SC	Subordinate conjunction
28	SCK	SC-Kar
29	SCP	SC pre-sentential
30	INJ	Interjection
31	PRT	Common particle
32	VALA	Vala particle
33	SYM	Common symbol
34	PU	Punctuation symbol
35	FF	Foreign fragment

3.1.2 Phrase Labels

Phrase labels are used to annotate constituents by providing a constituent label and phrasal boundaries. Constituents can also have sub-phrases in them which further combine to produce parse trees. The parse trees are helpful to define the syntax and semantics of a sentence. An annotation scheme should have labels to annotate all syntactic categories of a language. In case of morphologically-rich nature of a language, phrase label set needs to be derived carefully. The second question asked by [148] is about the morphological influence on the phrase labels and POS tags for a morphologically-rich language. The POS tag set has been derived to be more flat as discussed in Section 3.1.1. The scheme for the annotation of the CLE-UTB has been derived from Penn Treebank [107] and a universal tag set [73] which has been presented in [87]. The Penn Treebank used the standard labels which include, S, NP, PP, ADJP, ADVP, VP etc. It has 26 phrase labels and 17 functional labels [16, 106]. It includes additional labels for wh-clauses and wh-words. These labels include WHNP, WHPP, WHADJP, WHADVP, SBARQ and SQ which results the treebank to have more labels. [73] proposed an optimal phrase label set by analyzing 25 treebanks for 21 languages. They achieved improved parsing accuracies by using the simplified annotation scheme. They combined the related and duplicate labels into single categories. For example, all types of NPs were combined to a single NP label and all types of clauses were assigned the 'S' label. They proposed a label set with nine phrase labels. Their label set helped to achieve better parsing results but it lost some linguistic information of the parse trees. To annotate the CLE-UTB we have proposed a phrase label set shown by Table 3.2.

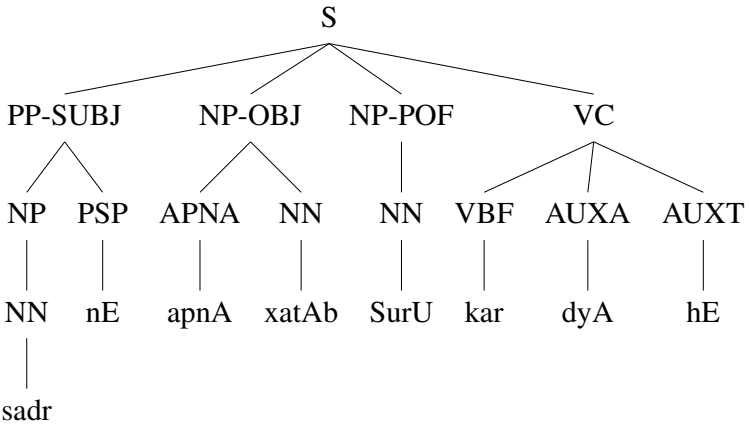
The phrase label set contains 11 labels which are mostly inherited from existing

TABLE 3.2: Phrase label set.

Sr#	Phrase label	Phrase category
1	NP	Noun Phrase
2	VC	Verb Complex
3	PP	Postpositional Phrase
4	ADJP	Adjective Phrase
5	ADVP	Adverbial Phrase
6	QP	Quantifier Phrase
7	S	Simple Clause
8	SBAR	Subordinate Sentence
9	PREP	Prepositional Phrase
10	DMP	Demonstrative Phrase
11	FFP	Foreign Fragment Phrase

label sets. The labels NP, ADJP, ADVP, QP, S and SBAR are used to annotate canonical constructions for noun phrases, adjective phrases, adverbial phrases, quantifier phrases, matrix and subordinate clauses. VC label has been used for verbal complex structure. In Urdu, a verbal structure contains a main verb followed by auxiliary verbs. All of these verbs and auxiliaries are annotated as a single constituent by using VC label. In Penn Treebank, PP label is used for prepositional phrase but Urdu has fewer occurrences of prepositions hence annotated by PREP label. However, the PP label has been used to mark post-positional phrases. Due to a strong case system, Urdu uses case markers which are represented by independent clitics. Demonstrative phrases annotate the constructions having demonstrative pronouns. Foreign fragment phrase has been used to annotated foreign words. Figure 3.1 shows a sample sentence annotated by this annotation scheme.

The annotated sentence in Figure 3.1 has been transliterated by using a Roman



sadr=nE apnA xatAb SurU kar dya hE
president.Sg=Erg own.M.Sg speech.M.Sg.Nom start do give.Perf.M.Sg be.Pres.Sg
The president has started his speech.

FIGURE 3.1: A phrase structure parse tree of a sample sentence .

transliteration scheme presented in [105]. The parse tree has a root label S which represents a sentence. There are four constituents under 'S' label with labels PP-SUBJ, NP-OBJ, NP-POF and VC. PP label marks a post-positional phrase which contains an NP followed by a post-position to represent an ergative case. An ergative case is represented by using the case marker 'nE'. The first NP phrase contains a common noun preceding by a reflexive pronoun *apnA*. The second NP contains a common noun *SurU* 'start'. The last constituent is labeled with a verb complex (VC) which has a main verb *kar* 'do' followed by aspectual and tense auxiliaries *dyA* 'give.Perf.M.Sg' and *hE* 'be.Pres.Sg' respectively. It can be seen that the parse tree also has functional labels which are attached with phrase labels. The labels SUBJ and OBJ represent subject and object of the sentence. The label POF marks the complex predicate structure. Section 3.1.3 describes the functional labels of the annotation scheme and Section 3.2 presents the annotation guidelines in more details.

3.1.3 Functional Labels

The annotation scheme further annotates a functional layer which represents the grammatical relations. The functional labels are concatenated with phrase labels with a hyphen '-'. Table 3.3 presents the list of functional labels which are part of our annotation scheme.

TABLE 3.3: Functional label set.

Sr#	Functional label	Grammatical function
1	SUBJ	Subject
2	OBJ	Object
3	OBL	Oblique
4	ADJ	Adjunct
5	POF	Part of function
6	PDL	Predicate link
7	VALA	VC-Vala
8	VOC	Vocative
9	INJ	Interjection
10	G	Genitive

Our annotation scheme used ten functional labels which have been derived from three sources, Penn Treebank, Urdu.KON-TB and HUTB. The labels SUBJ, OBJ and ADJ were derived from the Penn Treebank. SUBJ and OBJ labels mark subjects and objects. ADJ label defines adjuncts and spacio-temporal nouns. However, Penn Treebank and Urdu.KON-TB use separate tags to annotate spacial and temporal nouns. The Penn Treebank uses LOC and TMP labels for them whereas Urdu.KON-TB uses SPT and TMP labels. Our annotation keeps the label set compact and uses a single label ADJ

and marks them adjuncts. The label OBL has been used to annotate non-core compulsory arguments in the treebank. The label OBL has been annotated with NPs as well as PPs. The label POF annotates the complex predicate structure of Urdu and is attached with NPs, ADJPs and QPs. The label has been inherited from HUTB. Similarly, PDL label is attached with NPs, ADJPs, QPs and ADVPs to annotate copula constructions. The Urdu.KON-TB uses the tag PLINK to show the predicate link (PDL). The label 'G' annotates the genitive case markers along with post-positional phrases which is equivalent to POSS label of Urdu.KON-TB. The VOC label adopted from the Penn Treebank has been used to mark vocative relation. Furthermore, INJ and VALA labels have been used to annotate interjections and verb complex having *vAlA* particle. The *vAlA* particle usually appears after infinitive verbs, nouns and noun modifiers. The VALA functional labels has been used with VC label and other constituents which are annotated with phrase labels.

3.2 ANNOTATION GUIDELINES

The annotation guidelines produce phrase structure trees with a relatively flat structure for the CLE-UTB [87]. The annotation scheme has the ability to annotate sentences having any order of syntactic arguments and adjuncts. For that purpose, syntactic arguments and adjuncts are attached at the same level in a syntax tree. This mechanism produced the simplified trees without argument scrambling and traces in the treebank. This tree structure offers the dependency information within constituents and arguments in a sentence. The annotation guidelines have been largely derived from the Urdu.KON-TB which annotates non-binary trees providing deep linguistic information due to its

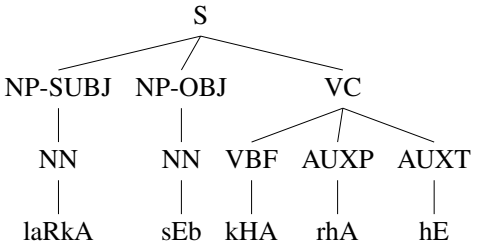
syntactically rich and large label set. Our treebank on the other hand uses a appropriate annotation label sets along with some language specific syntactic constructions including the annotation of complex predicates, genitive case markers, subordinate clauses and conjunctive participles.

3.2.1 Phrase Structure Annotation

In this section, we present the phrase structure annotation for different phrase labels with the help of examples.

3.2.1.1 Verb Complex

A verb complex (VC) of Urdu contains main verb, light verbs, copula and auxiliary verbs. The annotation scheme marks four auxiliary verbs which are aspectual (AUXA), modal (AUXM), progressive (AUXP) and tense auxiliary (AUXT). The auxiliaries have been categorized based on their POS tags. An Urdu verb complex may also contain negation (NEG) and emphatic particles. Figure 3.2 shows a sample sentence containing a nominal subject, a nominal object and a verb complex (VC). The VC phrase contains a finite verb with a POS tag VBF and two auxiliary verbs. The AUXP tag shows progressive action and the AUXT tag indicates the tense of the sentence. The POS tag of tense auxiliary remains the same for all tenses but lexical values indicate the actual tense in a sentence. An Urdu VC is head initial constituent because the main or light verbs are normally followed by auxiliary verbs.



laRkA	sEb	kHA	rhA	hE
boy.M.Sg.Nom	apple.M.Sg	eat	Prog.M.Sg	be.Pres.3.Sg

'The boy is eating an apple.'

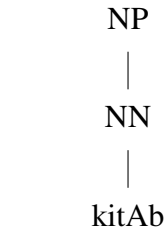
FIGURE 3.2: Annotation of a sentence containing a verb complex (VC), a nominal subject and a nominal object.

3.2.1.2 Noun phrases

The CLE-UTB marks all types of nouns including common nouns, proper nouns, pronouns, spacio-temporal nouns and noun complex predicates (Noun + verb) in a noun phrase (NP). An NP also contains modifiers, determinants and intensifiers. The POS tag set marks spacio-temporal and nominal complex predicate nouns with a single tag NN. This reduces the size of the POS tag set and all these types of nouns are annotated in an NP. The functional labels are used to annotate grammatical relations by adding an additional layer on noun phrases. Figure 3.3 shows some examples of noun phrases.

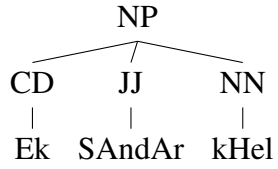
In Urdu, *ezafe* constructions constitute noun phrases with reverse order of nouns and their modifiers. A noun phrase normally contains one or more modifiers followed by a noun. In *ezafe* constructions, a noun appears before its modifier. These constructions use a vowel sound which is pronounced with head word appearing at the left-hand side of the constituent. Examples are given in Section 2.1.4. We have annotated *ezafe* constructions as noun phrases.

(a)



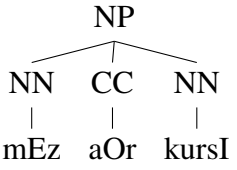
kitAb
book.F.Sg.Nom
'The book'

(b)



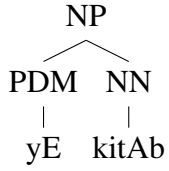
Ek SAndAr kHel
one splendid match.M.Sg.Nom
'A splendid match'

(c)



mEz aOr kursI
table.M.Sg.Nom and chair.F.Sg.Nom
'Table and chair'

(d)



yE kitAb
this book.F.Sg.Nom
'This book'

FIGURE 3.3: Examples of noun phrases (NPs)

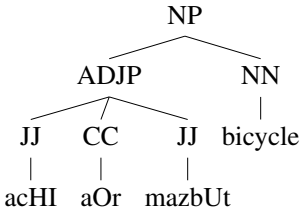
3.2.1.3 Adjective, Quantifier and Demonstrative phrases

Adjectives normally appear as noun modifiers but they also constitute an adjective phrase (ADJP) when there are multiple adjectives delimited by a coordinate conjunction (CC). Adjective phrases can also appear before verbs as part of complex predicates and copula constructions on pre-verbal positions. Their word order is flexible and adjective phrases do not always appear before verb construction. The Quantifier phrase is similar to ADJP as it is also a noun modifier. It also constitutes a quantifier phrase in case of complex predicates and copula constructions. A demonstrative pronoun (PDM) followed by a particle has been annotated as a demonstrative phrase (DMP). Figure 3.4 presents some examples of these phrases. Figure 3.4(a) shows an NP which has an ADJP containing two adjectives with a conjunction. Both adjectives modify head 'bicycle' of the NP. Figure 3.4(b) shows a copula construction with ADJP and it is marked with PDL functional label. It is important to note that copula constructions have also been annotated with NPs and QPs. Figure 3.4(c) shows a QP with copula construction and Figure 3.4(d) shows an NP containing a DMP which has a demonstrative pronoun followed by a particle *hI*.

3.2.1.4 Adpositional Phrases

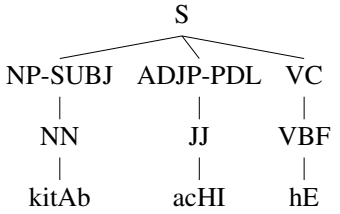
In the annotation scheme, a postpositional phase (PP) annotates the phrases containing case markers. A PP normally contains a noun phrase followed by a case marker. Urdu uses several clitics to represent case information. In the POS tag set, case markers are tagged with PSP label and prepositions are marked with PRE tag. The annotation scheme uses PREP label to annotate prepositional phrases in addition to PP label.

(a)



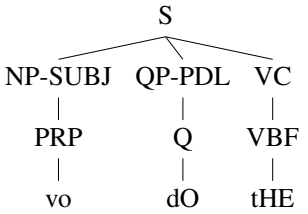
acHI aOr mazbUt bicycle
fine.F.Sg and durable bicycle.F.Sg.Nom
'A fine and durable bicycle'

(b)



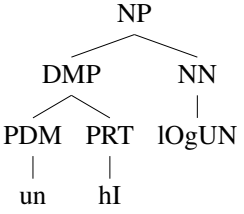
kitAb acHI hE
book.F.Sg.Nom fine.F.Sg be.Pres.3.Sg
'The book is fine.'

(c)



vO dO tHE
Pron.3.M.Pl two be.Perf.M.Pl
'They were two.'

(d)

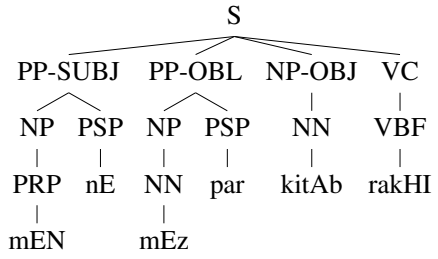


un hI lOgUN
Pron.3.Pl Prt.Intf people.M.Pl.Nom
'Those people'

FIGURE 3.4: Annotation examples (a,b) adjective phrase (ADJP) (c) quantifier phrase (QP) (d) demonstrative phrase (DMP).

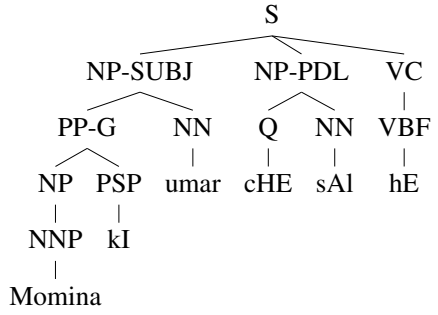
A PREP phrase contains a preposition followed by an NP. The grammatical roles are marked by using functional labels along with PP label. Ergative and dative subjects and accusative objects are marked by using SUBJ and OBJ labels respectively. The OBL label has been used to mark oblique constructions as shown by Figure 3.5(a). A PP containing genitive case marker is normally attached under an NP. Therefore, a functional label G is attached with inner PP to mark genitive case as shown by Figure 3.5(b).

(a)



mEN=nE mEz=par kitAb rakHi
Pron.1.Sg=Erg table.M.Sg=Loc book.F.Sg.Nom put-Perf.F.Sg
'I put the book on the table.'

(b)

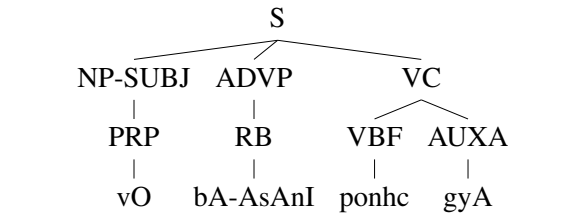


Momina=kI umar cHE sAl hE
Momina.F.Sg=Gen age.F.Sg.Nom six year.M.Sg.Nom be-Pres.3.Sg
'Momina's age is six years.'

FIGURE 3.5: Post-positional phrase (PP) annotation examples (a) ergative and locative case markers (b) genitive case marker.

3.2.1.5 Adverbial phrases

An adverbial phrase (ADVP) consists of adverbs and their combinations with particles. An adverb is the head word of the ADVP. Figure 3.6 shows a parse tree containing an ADVP where word *bA-AsAnI* 'easily' modifies the verb of the sentence. An ADVP can also contain negation and particles along with adverbs. The adverb in the example can be realized as *AsAnI sE* 'with ease' which constitutes a post-positional phrase. All such adverbial clauses with post-positions are annotated as PPs due to similar grammatical structure.



vO bA-AsAnI pohnc gyA
Pron.3.M.Sg easily reach go-Perf.M.Sg
'He reached easily.'

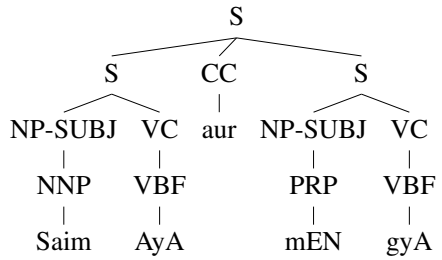
FIGURE 3.6: Adverbial phrase (ADVP) example.

3.2.1.6 Conjunctions

Coordinate conjunctions separate two or more words in a constituent (Figure 3.7). They also define relations between clauses. In Figure 3.7(a), subclauses have been annotated at the same level in a sentence under the label 'S'. Both elements have a coordinate conjunction *aOr* 'and'. Subordinate conjunctions have been used to annotate subclauses as dependents of independent clauses. The phrase label SBAR has been used to mark subordinate clauses as shown by Figure 3.7(b). A subordinate clause normally contains

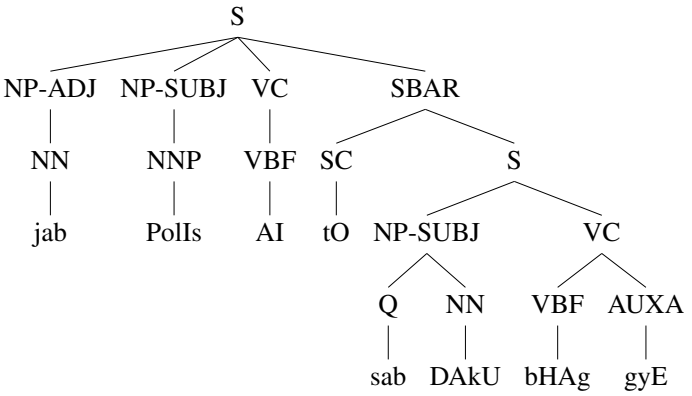
an ‘S’ clause preceded by subordinate conjunction clitic or punctuation. In the example of Figure 3.7(b), the Urdu sentence presents a subordinate conjunction, however its English translation turns the main clause into a subordinate clause.

(a)



Saim AyA aur mEN gyA
Saim.M.Sg.Nom come-Perf.M.Sg and Pron.1.M.Sg go-Perf.M.Sg
‘Saim came and I went.’

(b)



jab polIs AI tO sab DAkU
when police.F.Sg.Nom come-Perf.F.Sg then all people.M.Pl.Nom
bHAg gyE
run go-Perf.M.Pl
‘When police came then all the robbers ran away.’

FIGURE 3.7: Conjunction clause examples (a) coordinate conjunction (b) subordinate conjunction.

3.2.1.7 Non-core Clauses

Infinitive and relative clauses appear without any kind of relative pronouns in a sentence. These clauses have been annotated with ‘S’ label. These clauses may appear

under a noun phrase and a post-positional phrase as modifiers. For an NP, they are noun modifying clauses and for a PP they mostly appear as infinitive clauses. In PPs, these clauses are followed by case markers. At sentence level, they sometimes constitute core arguments and are labeled with appropriate functional tags. Figure 3.8(a) shows a parse tree with conjunctive participle having a word *kar* 'do' with POS tag SCK. In the POS tag set, this tag is assigned to only two words *kar* and *kE*. *kar* represents the completion of a task before the actual action in a sentence. These clauses have been annotated under SBAR label. Figure 3.8(b) shows a parse tree with an infinitive clause as a noun modifier hence attached under NP.

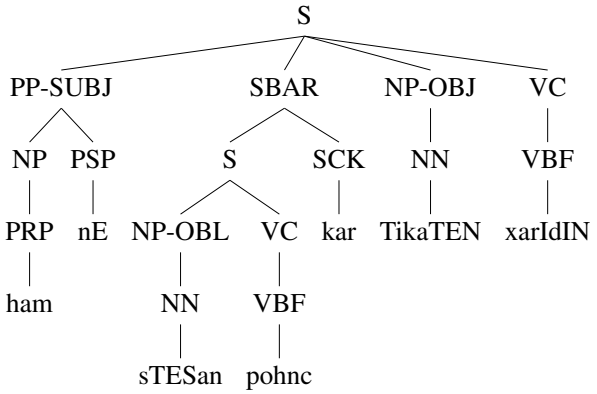
3.2.1.8 Foreign Fragment phrase

The foreign fragment phrase (FFP) is similar to noun phrase but it contains foreign words rather than nouns. It normally occurs within a noun phrase when a phrase of foreign language words is in parenthesis or quotes. FFPs are annotated in a flat structure which means all foreign words are annotated at a same level with flat structure and they have left-most token as head. The words from foreign languages which are not written in the Urdu script are marked as foreign fragments using the FF tag. We annotate all these fragments in FFP to keep their annotation simple and consistent.

3.2.1.9 Functional labels

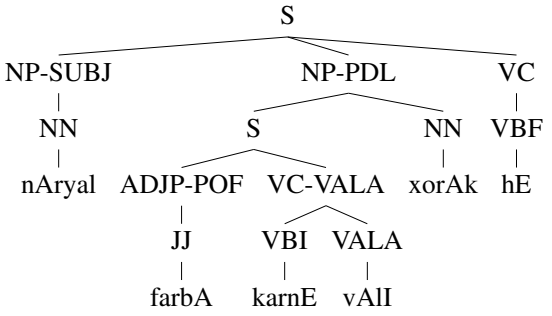
The annotation scheme uses functional labels to represent grammatical relations in parse trees. Ergative, nominative and dative subjects are marked with SUBJ label. The clitic 'nE' represents the ergative case which is annotated as a post-positional

(a)



ham=nE sTESan pohnc kar TikaTEN xarIdIN
Pron.1.Pl=Erg station.M.Sg.Nom reach do ticket.F.Pl.Nom purchase-Perf.F.Pl
'We purchased tickets after reaching the station.'

(b)



nAryal farbA karnE vAlI xorAk hE
Coconut.M.Sg.Nom healthy do-Inf.M.Pl Prt diet.F.Sg.Nom be-Pres.3.Sg
'Coconut is a fatty food.'

FIGURE 3.8: Annotation examples of non-core clauses.

phrase. Accusative and nominative objects are marked with OBJ label. Figure 3.9 shows parse trees with different functional labels. In Figure 3.9(a), the first NP contains a genitive case and has been marked with the G label. The POF label represents a complex predicate construction in which the head word *elAn* 'to announce' represents the action in the sentence. It is not a verb rather a noun producing the meaning of an action in combination with a light verb *karE* 'do'. In this example, the noun *elAn* along with verbal phrase make a complex predicate structure. In most cases, these kind of nouns,

adjectives and quantifiers have a pre-verbal position but it is not always the case. Our annotation scheme marks POF constructions as separate constituents so that they can be annotated independently of their position in a sentence. However, their dependency remains at the head of a sentence which is normally a verbal complex.

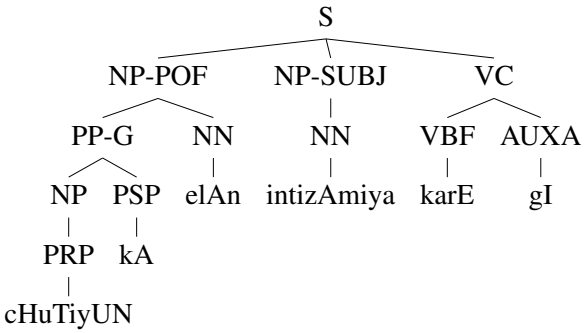
Non-core obligatory arguments have been annotated as oblique with OBL label. The OBL has also been used to mark non-canonical secondary arguments. Spacio-temporal noun phrases which behave as adverbial clauses are marked with ADJ label. However, spacial case markers have been annotated as post-positional phrases.

Figure 3.9(b) shows an example of a vocative relation which is annotated with a VOC label. Vocative label represents an addressing action in a sentence. The INJ has been used to mark interjections which show emotions, greetings and non-speech representations as shown by Figure 3.9(c). Another functional label which is used to mark verb complex is the VALA label. It marks a VC having a *vAlA* particle following main infinitive verbs. Although, the *vAlA* particle is also part of a noun phrase when it appears after a noun. In the annotation scheme of the CLE-UTB, functional labels represent grammatical relations between different elements in a sentence.

3.2.1.10 Ellipsis

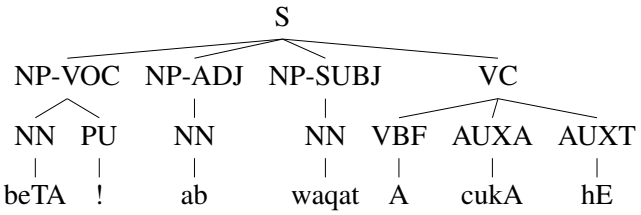
The noun and verb ellipsis exist without phonological content and are understood from contextual clues. The CLE-UTB annotates sentences without adding delta nodes which results in simplified trees. Figure 3.10 shows annotations of empty noun phrases and empty verb constructions.

(a)



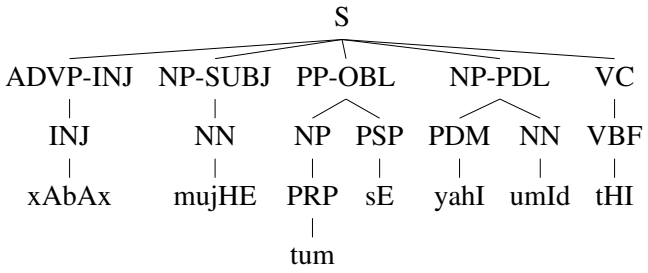
cHutiyUN=kA elAn intizAmiya karE-gI
holiday.F.PL=Gen announcement.M.Sg.Nom administration.F.Sg.Nom do-Fut.F.Sg
'The administration will announce holidays.'

(b)



beTA ab waqat A cukA hE
son.M.Sg=Voc now time.M.Sg.Nom come Pref.M.Sg be-Pres.3.Sg
'Son! the time has come now.'

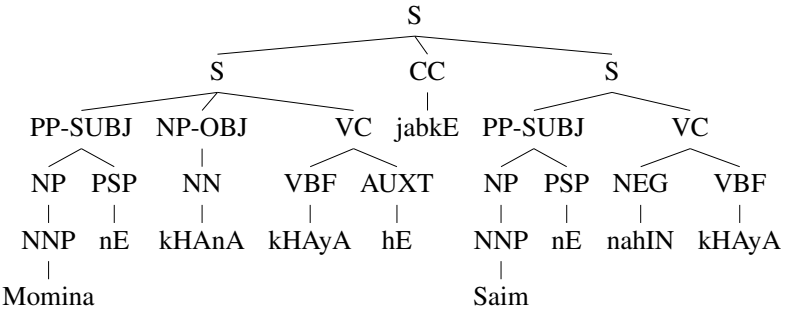
(c)



xAbAx mujHE tum=sE yahI umId tHI
nice=Inj Pron.1.Sg=Dat Pron.2.Sg=Abl this hope.F.Sg.Nom be.Pres.3.Sg
'Nice! This is what I expected from you.'

FIGURE 3.9: Annotation of functional labels (a) complex predicate (POF) (b) vocative (VOC) and (c) interjection (INJ).

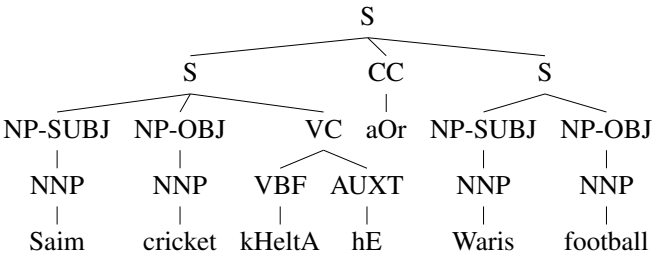
(a)



Momina=nE	kHAnA	kHAyA	hE	jabkE	Saim=nE
Momina.F.Sg=Erg	meal.M.Sg.Nom	eat-Perf.M.Sg	be.Pres.3.Sg	whereas	Saim.M.Sg=Erg
nahIN	kHAyA				
no	eat-Perf.M.Sg				

'Momina has eaten the meal whereas Saim has not.'

(b)



Saim	cricket	kHeltA	hE	aur
Saim.M.Sg.Nom	cricket.F.Sg.Nom	play-Pres.Hab.M.Sg	be-Pres.3.Sg	and
Waris	football			
Waris.M.Sg.Nom	football.M.Sg.Nom			

'Saim plays cricket and Waris football.'

FIGURE 3.10: Annotation of ellipsis (a) empty noun phrase (b) empty verb complex.

In Figure 3.10(a), the sentence has a noun ellipsis in the coordinate clause. The ergative subject and verb complex are present but nominal object is empty. The object information is understood from the first clause. Similarly, the parse tree in Figure 3.10(b) shows an ellipsis of verb complex in a coordinate clause. There is a missing verb complex and our annotation leaves it empty without adding extra nodes in the trees.

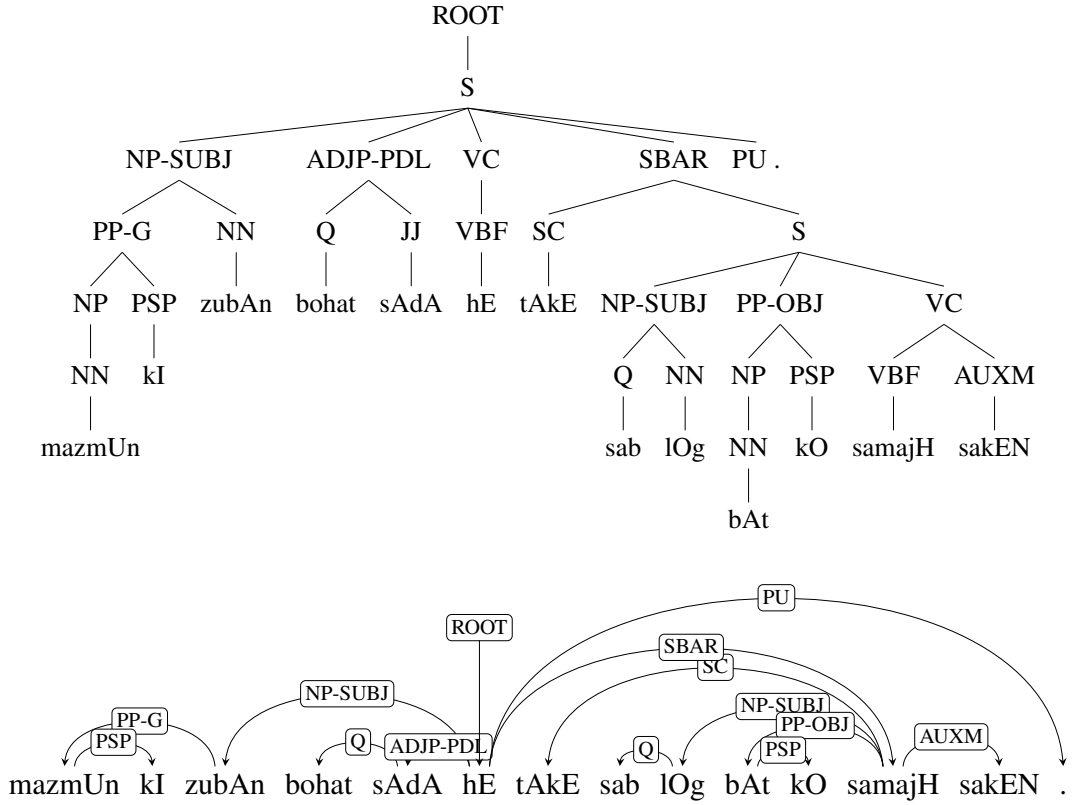
Our treebank has a simplified and a smaller annotation scheme, but it annotates

essential syntactic structure of the language. Our annotation scheme is also compatible with the dependency structure. Section 3.2.2 describes the process of annotation mapping from phrase structure to dependency structure.

3.2.2 Compatibility with Dependency Structure

A phrase structure parse tree represents the information of constituents and their grammatical relations in a sentence. A dependency structure on the other hand, provides the dependency information at the level of words by using dependency labels. The applications of natural language understanding require the information of dependency relations. The dependency structure provides the structure at word level and their contribution in the whole sentence. The annotation of the CLE-UTB has abstract phrase structure annotation by using a relatively flat label set along with grammatical relations which make it compatible to the dependency structure. The structure is helpful to employ an automated procedure which converts the phrase structure trees to dependency structure without manual annotation. Figure 3.11 presents a dependency structure parse tree against a phrase structure tree.

The phrase structure tree in Figure 3.11 presents the annotation of a sentence having a subordinate clause. The matrix clause has a nominal subject and a copula construction whereas the subordinate clause contains a nominal subject and an object represented with an accusative case followed by a verb complex. This phrase structure can be converted to dependency structure after identifying head words for each constituent. The grammatical relations can be used as dependency labels as shown by dependency structure in Figure 3.11.



mazmUn=kI zubAn bohat sAdA hE tAkE sab
article.M.Sg=Gen language.F.Sg.Nom very simple.M.Sg be.Pres.3.Sg SC all
lOg bAt=kO samajH sakEN
people.M.Pl.Nom content.F.Sg=Acc understand-Impr.M.Sg can.Pl
'The language of the article is very simple so that everyone can understand its content.'

FIGURE 3.11: A phrase structure parse tree by using the CLE-UTB analysis and its dependency representation.

At the initial step, the verb complex of the matrix clause is converted to root of the sentence. The nominal subject of the independent clause *zubAn* 'language' has a dependency on the root with a label NP-SUBJ. Similarly, the genitive case which is represented by the label PP-G has a dependency on the head word *zubAn* and the case marker 'kI' has a dependency on the word *mazmUn* 'article'. The predicate link annotated by the label ADJP-PDL shows a dependency on the root as well. The subordinate clause has a dependency on the root with a label SBAR. The main constituents of the subordinate clause are dependency on its main verb *samajH* 'understand'. The subject

lOg ‘people’ shows a dependency by using a label NP-SUBJ and the subject with label PP-OBJ. The leaf level constrictions mark their dependencies by using POS tags. Sentence end marker shows the dependency relation on the root. The dependency tree shows labels from our annotation scheme. An accurate head finding method along with the label mapping would be able to accurately convert the phrase structure tree into dependency structure. Figure 3.12 presents an equivalent dependency tree of tree from Figure 3.11 by using universal dependency[2] labels.

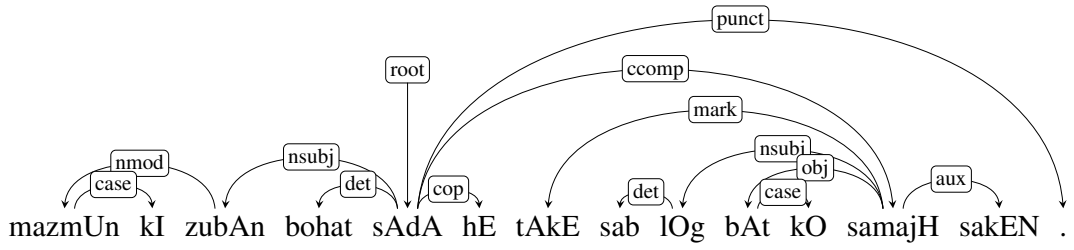


FIGURE 3.12: An equivalent dependency tree by using universal dependency labels.

A noun phrase has right most noun as head word. The PP phrases contain an NP followed by case markers. Therefore, they have NPs as heads. The head word for ADJP is also the right most word. The head word for VC is left most verb. VCs contain main verbs, finite or non-finite, which are followed by auxiliary verbs. Finite and infinitive verbs are tagged with different POS tags VBF and VBI respectively. The phrase structure labels shown in Figure 3.11 can be mapped on universal dependency labels deterministically. The label PP-G can be mapped on `nmod` (nominal modifier) which shows a genitive case. The label NP-SUBJ, representing the nominal subject, will be mapped on `nsubj`. The PP-OBJ which is the part of subordinate clause, will be mapped on `obj` dependency label. ADJP will be mapped on `amod` (adjectival modifier). The matrix clause has a copula construction marked with the label ADJP-PDL. The

[2]https://universaldependencies.org

PDL label can be used to find copula constructions and the root or the verb of a clause can be updated as shown in Figure 3.12. The dependency tree has *sAdA* 'simple' as root and the verb *hE* shows a copula dependency on the root. After devising a head model for Urdu and label mapping from phrase to universal dependencies, we can convert the CLE-UTB to dependency structure. Appendix A presents the initial conversion of our treebank to dependency structure and dependency parsing results.

3.3 CORPUS SELECTION AND PREPARATIONS

A balanced corpus is important for the annotation of linguistic resources. In this section, we discuss the selected corpus and its preparation for the annotation process.

3.3.1 Corpus

The CLE-UTB has been annotated by using a part of the CLE Urdu Digest corpus [150] [3] [4]. The corpus has been POS tagged by using a POS tag set developed in [86]. It is a balanced corpus and has text from fifteen text genres. Table 3.4 shows the number of sentences and tokens with respect to different genres.

The segmentation for sentences was performed manually because the sentence end markers were absent for a number of sentences. The corpus has sentences with different lengths and average sentence length is 18.9 tokens. Figure 3.13 further shows number of sentences with respect to number of tokens in them.

[3] http://www.cle.org.pk
[4] https://urdudigest.pk

TABLE 3.4: Number of sentences and tokens against different text domains of the corpus.

Sr#	Domain	No. of Sentences	No. of Tokens
1	Book reviews	390	8,676
2	Culture	421	8,131
3	Education	267	5,828
4	Entertainment	248	5,047
5	Health	534	9,594
6	Interviews	577	11,680
7	Letters	650	11,267
8	Novels	551	10,433
9	Press	424	9,779
10	Religion	488	9,354
11	Science	405	8,294
12	Short stories	1,664	27,563
13	Sports	505	9,933
14	Technology	124	2,464
15	Translation of foreign languages	606	10,532
	Total	**7,854**	**148,575**

3.3.2 Preprocessing

Preprocessing step includes cleaning and POS tagging. Extra and duplicate punctuations and symbols were removed from the corpus. Further preprocessing was performed in two steps. At first step, POS verification and sentence boundaries were identified manually. An existing Urdu POS tagger[5] was used which is based on the well-known Tree Tagger [133]. The POS verification helped to achieve accurate and consistent tagging throughout the corpus. The corpus was further divided into batches

[5] www.cle.org.pk/clestore/postagger.htm

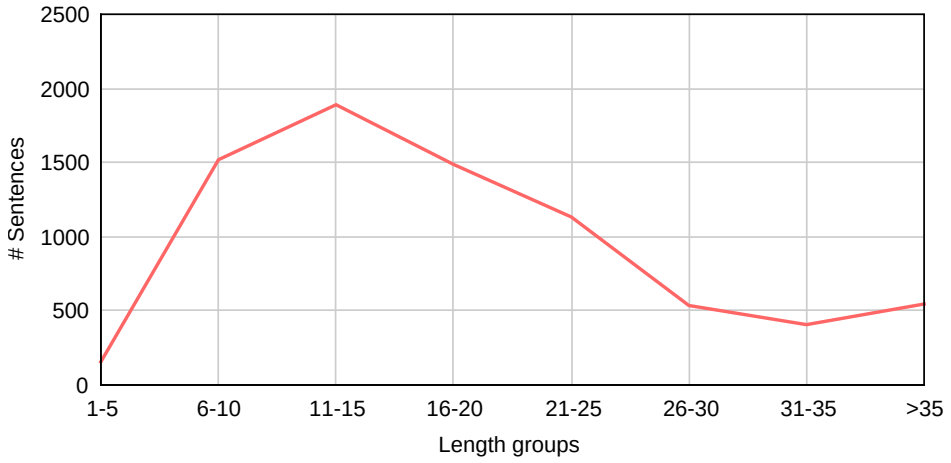


FIGURE 3.13: Corpus coverage according to number of tokens. No. of sentences are on y-axis and length groups are on x-axis.

for annotation. At the second step, unique sentence numbers were assigned to each sentence. The annotation was performed in the form of XML trees therefore, the number was attached as an attribute to the 'S' label.

During the annotation, several assisting utilities were developed and employed for the evaluation of the annotated corpus. These processes are as follows:

- *Reference Corpus Selector:* This process selected the random corpus referred to compute the inter-annotator agreement.

- *Grammar Extractor:* Grammar extraction has been performed for the grammar-based consistency evaluation. This process performs in two steps. In first step, all grammar rules were extracted with their frequencies. Infrequent rules have been reviewed with respect to the annotation guidelines and implausible rules were extracted. The second step takes the implausible rules as input and generates a report showing the file names and sentence numbers which contain any of

those rules. The sentences pointed by the report were further reviewed to find the annotation errors. Section 3.4.2.2 discusses the evaluation process in detail.

- *Automatic Consistency-Checker:* This tool evaluates the annotation consistency with respect to the contexts of constructions. It has been used to evaluate the treebank after the annotation was completed. Section 3.4.2.3 presents the details of the consistency checker and its results.

- *Testset Generation:* This process has been developed to select test sentences from the treebank. The sentences have been selected randomly from each domain with respect to number of sentences in domains and authors. It helped to produce an unbiased test data set.

- *XML to Brackets:* This process takes XML trees and converts them to bracket notation. The process has been employed for the training and testing of statistical parsers.

3.4 TREEBANK ANNOTATION EVALUATION

Accuracy and annotation consistency are crucial for the development of a linguistic resource. Therefore, multiple evaluation tasks have been performed to evaluate the treebank annotation. After completion, each batch was reviewed by a second annotator and the types of errors were communicated to the first annotator so that other batches would have less inconsistencies. These evaluation tasks are described in the following sections.

3.4.1 Completeness and Correctness Checking

The completeness evaluation task identified empty and incorrect POS and bracket labels. Phrase and functional labels both were checked in this process. Table 3.5 shows a sample report generated by this task and evaluation details are as follows:

TABLE 3.5: Completeness and correctness evaluation sample reports for (a) list of phrases with no content (b) list of incorrect POS tags (c) list of incorrect phrase labels.

(a)

Sr#	File name	Sentence no.	Rule
1	100K_562.xml	CLE_100K_TB_S202	NP->
2	100K_562.xml	CLE_100K_TB_S203	NP-POF->
3	100K_562.xml	CLE_100K_TB_S205	NP-SUBJ->
4	100K_563.xml	CLE_100K_TB_S231	ADJP-POF->
5	...		

(b)

Sr#	File name	Sentence no.	Incorrect POS
1	100K_562.xml	CLE_100K_TB_S202	AUXTF
2	100K_562.xml	CLE_100K_TB_S205	PDM.
3	100K_562.xml	CLE_100K_TB_S204	NNs
4	100K_562.xml	CLE_100K_TB_S204	VBFrahA
5	...		

(c)

Sr#	File name	Sentence no.	Incorrect label
1	100K_562.xml	CLE_100K_TB_S214	NP-SUB
2	100K_562.xml	CLE_100K_TB_S215	PP-GP
3	100K_562.xml	CLE_100K_TB_S216	NP-OBJP
4	100K_563.xml	CLE_100K_TB_S232	PP-SUB
5	...		

- *Labels lacking text:* The annotators marked phrase labels on the POS tagged Urdu text and some of the constituents were mistakenly skipped with no context in them. These types of errors produce empty phrases and only right-hand side of the grammar rules as shown in the Table 3.5(a). The errors identified in this process were manually corrected.

- *POS errors:* The annotation of POS tags were done by using a slash (/). If slash delimiter is missing or tokens are marked with erroneous tags, this process reports it with the concerned sentence and file name. All these type of errors were corrected after reviewing the sentences manually.

- *Bracketing errors:* This process identified erroneous bracketing labels which included both phrase and functional labels. It identified file names, sentence numbers and labels which were further corrected manually based on this report.

The completeness evaluation ensures that each token is assigned with a valid POS tag and phrase labels are correct. The process evaluates human errors in case of label annotation. However, if a valid POS tag is used to mark wrong word or a correct phrase label has been used to annotate a wrong constituent then these mistakes were not identified at this stage. The whole corpus was reviewed manually after the completion of each batch which was helpful to refine the annotated corpus. It is still possible that the corpus may have some inconsistencies. To identify annotation inconsistencies, we further employed evaluation methods as described in Section 3.4.2.

3.4.2 Consistency Evaluation

To evaluate the annotation consistency, we have computed inter-annotator agreement for random sentences from annotators. We devised a grammar-based evaluation method which have been used to evaluate the whole treebank for phrase labels as well as POS tags. Finally, an automatic treebank consistency checking tool has been employed to find outliers and annotation inconsistencies. The descriptions and evaluations of these evaluation methods are as follows.

3.4.2.1 Inter-Annotator Agreement

We have used an existing POS tagger for Urdu to accelerate the POS annotation process. However, the annotators corrected the incorrect tags manually during the annotation. The inter-annotator agreement has been computed for POS tags, phrase and functional labels. For that purpose, random sentences were selected for both annotators. The first annotator has done masters degree in linguistics and the other annotator is a doctoral students in computational linguistics. To evaluate brackets from both annotators, we have used parseval evaluation measures [5]. The measures helped to compute the annotation similarities in the form of f-scores. Table 3.6 presents the inter-annotator agreements.

Table 3.6 also shows an unseen corpus of two hundred sentences which was collected separately and annotated by both annotators independently. The agreement scores are given with respect to samples but overall phrase structure annotation scores are above 90% which are acceptable [152, 116]. The agreement for POS tags is 98.4% which is quite high. the agreement score for functional labels is 89.1%. We further

TABLE 3.6: Inter-annotator agreement against POS tags, phrase and functional labels.

Category	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Sample 6	Overall
POS	98.9%	98.8%	99.2%	98.5%	97.2%	98.4%	98.4%
Phrase labels	93.8%	90.3%	93.6%	90.3%	91.4%	93.9%	92.7%
Func. labels	90.5%	89.9%	89.6%	86.6%	85.9%	89.9%	89.1%
# Sentences	36	36	40	105	40	200	457
# Tokens	701	670	756	1,896	644	4,447	9,114

analyzed these scores against annotation categories with respect to top disagreements as shown by Table 3.7.

The agreement for POS tagging is 98.4%. Most of disagreements between both annotators are between demonstrative person pronouns (PDM) and personal pronouns (PRP). A demonstrative pronoun specifies a common noun which appears right after it. However, both pronouns are drawn from same set of words and therefore, cause confusions sometimes. The second most highest POS disagreement is between finite verbs marked with VBF tag and aspectual auxiliary marked with AUXA. The reason of this disagreement is again the lexical similarities. The POS tagging scheme marks the main verb as VBF or VBI which is followed by auxiliary verbs. Both of these types of verbs have different lexical values within a single verb complex (VC) but in the corpus, a main verb in one verbal construction can become an aspectual auxiliary in some other construction and vice versa. For example, the phrase *kar liyA* 'has done' presents a verbal complex with *kar* as main verb with tag VBF and *liyA* as aspectual auxiliary with tag AUXA. In another verbal construction *liyA gyA* 'has taken', the main verb is *liyA* which is followed by the auxiliary verb *gyA*. These types of lexical similarities in the corpus lead to some tagging confusions.

Another high disagreement was between adjectives (JJ) and common nouns (NN). The disagreement is again due to lexical similarities as many words are marked by these two tags based on their context. For instance, the word *fOjI* 'soldier' has NN tag and is marked as common noun when appears independently. When the word appears as the part of a phrase like *fOjI tAqat* 'military power', then it is the noun modifier and has the tag of an adjective. In the same way, there are disagreements between subordinate conjunctions and coordinates conjunctions with subordinate conjunction presentational (SCP). The tag SCP marks a subordinate conjunction item when it appears at the start of a sentence in the annotation scheme. However, its lexical value belongs to the set of other conjunctions. The lexical similarity causes the confusion with other conjunctions. It is quite clear that most of the disagreement are due to lexical similarities. The automated consistency evaluation methods are further employed to overcome such inconsistencies. Beside these disagreements, there were several random differences in the POS tagging but the accumulative POS agreement is quite satisfying.

The agreement for phrase annotation is 92.7%. The highest disagreement is between adjective phrase (ADJP) and noun phrase (NP). According to the annotation guidelines, an adjective phrase is annotated when there are more than one noun modifiers separated by a coordinate conjunction or the phrase is making a copula construction or it is part of the complex predicate structure. However, copula and complex predicate constructions are marked by using functional labels to differentiate their role in a parse tree. The reason behind this disagreement is the disagreement of POS tags due to lexical similarities. The second most disagreement is between NPs and quantifier phrases (QP). A quantifier is a noun modifying phrase and it is also part of copula

TABLE 3.7: Top ten disagreements with respect to POS tags, phrase and functional label annotation.

POS			Phrase Labels			Functional Labels		
Ann.1	Ann.2	Count	Ann.1	Ann.2	Count	Ann.1	Ann.2	Count
PDM	PRP	14	NP	ADJP	8	SUBJ	OBJ	14
PRP	PDM	13	NP	QP	5	OBJ	SUBJ	12
VBF	AUXA	13	QP	NP	5	OBJ	OBL	9
NN	JJ	10	NP	ADVP	4	POF	SUBJ	8
JJ	NN	10	ADJP	NP	4	POF	OBJ	7
NN	NNP	8	VC	ADVP	2	POF	PDL	5
AUXA	VBF	8	NP	PREP	1	SUBJ	PDL	4
SCP	SC	7	PP	NP	1	PDL	SUBJ	4
NNP	NN	6	S	VC	1	OBJ	POF	4
CC	SCP	4	NP	S	1	PDL	POF	3

and complex predicate structure. The reason for these disagreement is also the close lexical resemblance and POS disagreements which is the main reason that most of the disagreements are two sides. There are further disagreements between NPs and adverbial phrases (ADVP). The annotation guidelines mark interjections as ADVPs. There were POS mismatches where interjections were tagged with common noun tag which resulted in the NP constructions. There are disagreements between adverbial phrase and verb complex structure. After the analysis, these disagreements came up again due to POS mismatches. The adverbs were marked as verbs resulting them to become part of verbal complex. There are also some arbitrary phrase label mismatches but the overall phrase label agreement score is quite acceptable for a phrase structure treebank.

The overall agreement for functional labels is 89.1%. Table 3.7 also presents the

top 10 disagreements between the annotators. The most disagreements are between subjects and objects in both directions. In the annotation guidelines, a subject is annotated as a nominal subject or as an ergative/dative subject. Similarly, objects are annotated as nominal objects or accusative objects. The first reason behind these disagreements was observed the absence of direct subjects or objects and pronoun drops in the passive declarative sentences. The second reason is ellipsis as the annotation scheme does not annotate ellipsis as external labels. These kind of annotations created confusions hence resulted in disagreements. The third most disagreements are between objects and oblique labels. The oblique (OBL) labels mark the non-core compulsory arguments and recipients. The label OBL is attached with noun phrases as well as post-positional phrases with the dative case marker *kO*. An accusative case is also represented by the same clitic which created confusions. The functional labels for subject and object were also confused with the part of function (POF) and predicate link (PDL) labels. These two constructions usually have pre-verbal positions and sometimes make a sense of agent or object resulting in disagreements. Both part of functions and predicate links are annotated along with noun, adjective and quantifier phrases. This annotation property also created some confusions between POF and PDL labels. Ellipsis and passive sentences were also influential to some annotation disagreements. However, overall agreement for functional labels is acceptable. Following sections describe annotation consistency evaluations which were helpful to correct the annotation errors.

We have further calculated the Kappa coefficients for inter-annotator agreements by computing Cohen's Kappa [35]. The Kappa coefficients represent the agreements

between annotators and raters. The coefficients have been computed against POS tagging, Phrase labeling and constituents. Table 3.8 shows the coefficient values computed during the annotation of the CLE-UTB.

TABLE 3.8: Kappa coefficients for inter-annotator agreements against POS tagging, constituent labeling, constituents based on the tokens in the reference corpus.

POS tagging	Constituent labels	Constituents	Labeled constituents
0.982	0.904	0.815	0.791

The interpretation of Kappa coefficients is crucial to demonstrate the agreements between annotators. The values between 0.01 and 0.20 are interpreted as poor agreement, the values from 0.21 to 0.40 are considered as fair agreements, the score between 0.41 and 0.60 depicts the moderate agreement, the values from 0.61 to 0.80 show a substantial agreement and the coefficient values from 0.81 to 1.0 present almost perfect inter-annotator agreement [109]. The score for POS tagging is 0.982 which depicts a perfect agreement between annotators. The coefficient for phrase labels is 0.904 which is also quite high. The kappa scores for constituents and labeled constituents are 0.815 and 0.791 respectively. These agreement scores are acceptable as suggested in [109].

3.4.2.2 Grammar-based Consistency Checking

We devised a semi-automated process to perform post-annotation consistency evaluation. Our grammar-base consistency checking mechanism, identifies the linguistically implausible constituents which are reviewed and updated according to the annotation guidelines. It finds the implausible constructions based on their frequencies in the treebank. The constructions having higher frequencies are considered to be correct and linguistically plausible. The constructions with low frequencies and rare occurrences

TABLE 3.9: A sample context-free grammar with rule frequencies sorted in non-decreasing order.

Rule #	Grammar rule	Frequency
1	NP-SUBJ→PDM JJ PRT NN	1
2	S→PP PP-OBL NP-SUBJ NP-POF VC PU	1
3	ADJP→APNA JJ JJ JJ	1
4	S→NP-ADJ NP-SUBJ NP-ADJ PP-OBL VC SBAR PU	1
5	S→NP-SUBJ SBAR NP-ADJ NP-POF VC	1
	...	
9489	NP→PP-G NN	3094
9490	PP→NP PSP	6162
9491	NP→NN	6593
9492	VC→VBF	7221
9493	PP-G→NP PSP	8222

need to be checked whether they are correct or not. In this evaluation task, phrase labels and POS tags were evaluated.

The evaluation process took the annotated parse trees as input and produced a context-free grammar. The grammar further contains rule frequencies. Each rule has a head which is a phrase label along with its functional label, rule body and its frequency. The rule body has a combination of phrase labels and POS tags. The frequency is a count of occurrences of that constituent throughout the treebank. The grammar for non-terminal and terminal constructions were evaluated separately. The terminal grammar contains POS tags as rule heads. At the first step, non-terminal grammar has been evaluated. Table 3.9 shows some most and least occurring rules from the extracted grammar.

The evaluation process was performed in two steps. In the first step, we prepared a list of rules which were potentially implausible. For that purpose, all grammar rules were reviewed starting from low frequent rules. The doubtful rules were put into an implausible grammar. Table 3.10(a) presents a sample implausible grammar. In some cases, the same constituent may appear under different phrase label. Therefore, for such rules, we put the question mark (?) so that all such constructions could be extracted for review irrespective of their rule head. However, the right-hand sides should not be empty. The second step, takes the implausible grammar as input and scans the whole treebank against each rule and generates a report containing file names, sentence numbers and grammar rules as shown in Table 3.10(b). The sentences in the generated report were reviewed manually and updated in case of incorrect annotations. It is important to note that all the reported parse trees were reviewed but not all the trees were updated because many constructions were found plausible during the review task.

To generate the report for potentially implausible constructions, the grammar contained 575 rules which were used in the second step. By using the implausible report, overall 1,397 annotated sentences were reviewed. The phrase and functional labels both were evaluated. Table 3.11 shows the revision statistics with respect to phrase and functional labels.

Table 3.11 shows the number of sentences against annotation labels. Most of the sentences were reviewed for NPs followed by PPs and VCs. Before performing the grammar-based evaluation, the context-free grammar contained 9,493 rules which were reduced to 9,170 after revision. The reduction is of 323 rules. However, this is not a big difference as the grammar has a large number of rules. The first reason

TABLE 3.10: (a) List of potentially implausible grammar rules prepared to extract sentences for revision (b) Evaluation report based on a list of grammar rules

(a)

Sr#	Rules
1	? → NP PRT PSP PSP
2	PP → NP PSP PSP PSP PRT
3	? → PRT NN
4	? → PRT
	...

(b)

Sr#	File name	Sentence no.	Grammar rule
1	file1.xml	CLE_500K_TB_S6	PP-OBL → NP PRT PSP PSP
2	file2.xml	CLE_100K_S373	NP-ADJ → PRT NN
3	file3.xml	CLE_100K_S198	PP → NP PSP PSP PSP PRT
	...		

is that the whole treebank was reviewed manually before grammar-based evaluation. Second, the sentences generated by the grammar report were reviewed but there were many constituents which were already annotated correctly therefore, not all sentences were revised. The context-free grammar considers each rule independent of each other. Therefore, there is a need for a method which evaluates the phrasal annotation with respect to their contexts. Section 3.4.2.3 discusses an automated consistency checking tool which uses contextual information to find implausible annotations.

The grammar-based consistency evaluation has been performed to check POS tagging by using terminal rules. Terminal rules usually contain POS tag as head and token as body. We have updated the grammar to find the tagging inconsistencies. A single token can have more than one POS tags according to its context and syntactic contribution. We counted how often each word occurred with each POS tag and sorted

TABLE 3.11: Statistics of sentences and labels which have been reviewed or revised based on implausible grammar report.

Phrase labels	Functional labels	# Sentences reviewed
NP	–	483
	SUBJ	211
	POF	37
	PDL	37
	OBJ	31
	ADJ	15
	VOC	10
	OBL	6
	INJ	3
PP	–	82
	G	43
	SUBJ	27
	OBL	25
	OBJ	6
	PDL	3
ADJP	–	44
	PDL	14
	POF	6
	SUBJ	2
	OBJ	2
ADVP	–	52
	INJ	3
	PDL	1
	POF	1
	VOC	1
QP	–	4
	SUBJ	2
VC	–	151
S	–	10
SBAR	–	81
PREP	–	3
FFP	–	1
Total	–	1,397

TABLE 3.12: Sample POS grammar containing POS tags against words and number of unique tags and statistics of words with count of POS tags.

Sample POS grammar		Word statistics with # POS tags		
Word	POS tags	Category	#Words	#Instances
aOr 'and'	**CC**,**SCP**,**JJ**,**Q**,PRT,AUXT	Six POS words	2	3,445
sI 'like'	**PRT**,**NNP**,**VBF**, JJ, NN	Five POS words	8	5,223
kI 'genitive'	**PSP**, **VBF**, **NN**, VBI	Four POS words	25	11,054
un 'pronoun'	**PRP**, **PDM**, NN	Three POS words	130	29,361
mujHE 'pronoun'	**PRP**, NN	Two POS words	876	24,566
kitAb 'book'	**NN**	Single POS words	13,074	74,926
–	–	Total	14,115	148,575

the tags of each word by frequency. The sample POS grammar and POS statistics of the treebank with respect to number of tags are shown in Table 3.12.

Table 3.12 shows a POS grammar in two forms. The form with word statistics with respect to number of tags presents the number of uniques words have six tags to one tag and there occurrences in the corpus. The sample POS grammar shows tokens and all assigned tags to them. For example, word *aOr* 'and' has been marked with six POS tags CC, SCP, JJ, Q, PRT and AUXT. The word *aOr* can have multiple senses according to its usage. Mainly, it is used for coordinate conjunctions and hence have CC tag. It also bears the meaning of 'different' and 'more' which are marked with tags JJ and Q. In the POS tagging scheme, if a coordinate or subordinate conjunction appears at the start of a sentences then it is marked with SCP tag. Hence, tags PRT and AUXT which are used to mark particles and tense auxiliaries have been used incorrectly. The incorrect tags for such tokens were reviewed and corrected throughout the treebank. In this process, 1,041 words were reviewed and POS tags were corrected. The grammar-based evaluation helped to identify annotation errors which were independent of their

contexts. The automatic consistency checker performed context-based evaluation as presented in Section 3.4.2.3

3.4.2.3 Automatic Consistency Checker

The grammar-based consistency evaluation was helpful to identify annotation consistencies by using context-free grammar rules. There is a need to evaluate treebank annotation by looking at the context of constituents to identify outliers and annotation inconsistencies. Therefore, we have further used an automated consistency checking tool[6] as presented in [82]. It creates groups of unique constructions with respect to annotation labels and computes the skewness of groups in the treebank. The severity of inconsistencies are depicted by skew values. A high skew-value represents a higher annotation inconsistency. The tool returns a report which contains group items, their skew values, annotation labels and corresponding sentence numbers in the treebank.

The skew-value is computed for each group with respect to frequency distribution of all constructions in the treebank. The formula to compute the skew-values is presented by Equation 3.1.

$$Skewvalue(g) = \begin{cases} \sum_{c \in C}(f_c - mean(f_C))^2 & if |C| > 1 \\ -1 & if |C| = 1 \end{cases} \tag{3.1}$$

Where g specifies a unique group, f_c is the frequency of a category, C represents all categories in the corpus, $mean(f_C)$ computes the statistical mean of frequencies and $|C|$ shows the total number of elements in C.

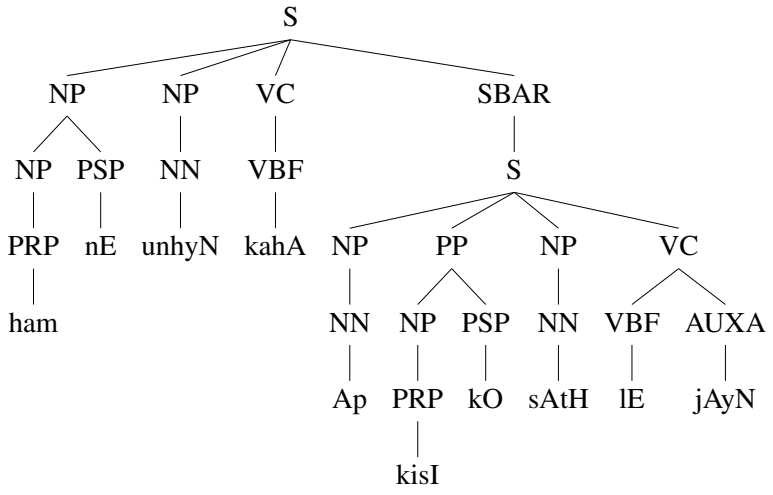
[6] https://github.com/Kaljurand/treebank-consistency-checking

TABLE 3.13: Sample results of consistency-checker for lexical items, POS tags and phrase labels.

Category	Group	Skew-value	Frequency	Phrase labels	Sentences no.
Lexical items	ham_nE	1404.5	54	PP	-
			1	NP	2519
POS	INJ_hAN	84.5	14	ADVP	-
			1	NP	5873
Phrase labels	NP_[sab]	264.5	24	NP	-
			1	S	3009

A skew-value represents the type and severity of inconsistencies for different categories. The categories can be created for lexical items, POS tags and phrase labels. The skew-value of -1 shows that a specific category has only one group and hence can be considered as consistent. Higher skew-values are used to identify annotation errors. The value near or equal to zero means that the group has almost even appearances in different categories. There are three main reasons of higher skew-values. First, the annotation of a group is flexible and has possibility to appear in different contexts. Second, difference of understanding between annotators which leads into annotation inconsistencies. Third, ambiguity in the annotation scheme which resulted the annotation in different categories randomly. All of these inconsistencies have been addressed and resolved by using the consistency checker. For that purpose, all groups with skew-values greater than zero were reviewed. The groups were further evaluated by analyzing on the basis of their contexts. Groups were crated for tokens, POS tags and phrase labels. Table 3.13 presents a sample evaluation result report for three types of groups. The table shows groups, their skew-values and frequency of occurrence under phrase labels for each category.

Sample groups have been shown for each category in Table 3.13. For example, the lexical group *ham_nE* ‘pron.1.Pl=Erg‘ has 54 appearances in PP and only one occurrence under an NP. It has a skew-value of 1404.5. The last column of the table shows sentence numbers if occurrences are less than ten. If a group has a frequency of ten or higher then it was considered as consistent. Figure 3.14 shows the parse tree against sentence number 2519. It is quite intuitive that the group annotated under NP needs to be reviewed as it seems implausible.



ham=nE unHyN kahA Ap kisI=kO sAtH lE jAyN
pron.1.Pl=Erg pron.3.Pl=Dat say-Perf.M.Sg pron.2.Pl pron.3.Sg=Dat with take go-Impr.M.Pl
We said to them that you should take someone with you

FIGURE 3.14: Parse tree for the sentence no. 2519 (from Table 3.13) in the treebank.

The second category shows POS tagging skewness. It shows the word *hAN* ‘yes’ which was annotated with a POS tag INJ under ADVP 14 times and under NP, only once. In these categories the skew-value was 84.5. Similarly, the third category computed the skew-value against an NP where *sab* ‘all’ is the next token. The group has a skew-value of 264.5 and has been annotated as NP 24 times and as ‘S’ only once. These inconsistencies can be reached by using the sentence numbers. In the evaluation process, all groups have been reviewed with skew-value of greater than zero.

The tool has been employed by choosing context of zero and one for all three categories. Table 3.14 presents the evaluation report with respect to number of sentences reviewed for inconsistencies.

TABLE 3.14: Groups and sentences reviewed against lexical items, POS tags and phrase labels for zero and one context.

Context	Tokens		POS tags		Phrase labels	
	#Groups	#Sentences	#Groups	#Sentences	#Groups	#Sentences
Zero	288	486	195	279	303	82
One	12	16	134	119	83	37

Table 3.14 shows number of sentences and groups which were reviewed during the evaluation. Overall, 502 sentences were reviewed against category of tokens, 398 sentences were reviewed for POS tags and 119 sentences were reviewed for phrase labels. The evaluation process was carried out in a sequential manner. Inconsistencies for tokens were checked for context zero and one before the analysis of POS tags and finally phrase labels were evaluated. Section 3.5 presents the treebank statistics of the final version of the treebank.

3.5 TREEBANK STATISTICS

This section presents label-wise statistics of the treebank. A few phrase labels have more frequency in the corpus as compared to others. Table 3.15 shows frequencies of annotation labels. Similarly, Table 3.16 presents the statistics of POS tags after evaluation.

The NP label has highest frequency in the treebank. The annotation scheme marks all types of nouns including common nouns, spacio-temporal nouns and complex

TABLE 3.15: Frequencies of phrase and functional labels in the final treebank.

Phrase labels	Frequency	Functional labels	Frequency
NP	48,359	SUBJ	10,507
VC	17,530	OBJ	4,067
PP	21,416	OBL	2,324
ADJP	3,331	ADJ	3,094
ADVP	2,114	POF	5,296
QP	1,136	PDL	3,061
S	20,791	VALA	213
SBAR	4,741	VOC	257
PREP	33	INJ	100
DMP	57	-	-
FFP	138	-	-
Total	119,646	Total	28,919

predicate structures (noun + verb) by using a single POS tags NN and hence with the same phrase label NP. Proper nouns are also annotated with NP label. In the same way, VCs have a prominent occurrences in the corpus. Each sentence usually has at least one verbal construction but sentences with subordinate and coordinate conjunction and non-core clauses, have additional verbal constructions which increased its frequency in the corpus. The case system of the language is also depicted by higher frequency of PP phrases. All types of case markers have been annotated as PP. The phrase label 'S' has been used to annotate all types of clauses and hence it has more occurrences even way higher than the total number of sentences. SBAR label has been used to annotate subordinate clauses and its frequency shows a sufficient representation in the corpus. A few phrases including PREP, DMP and FFP have lower frequencies due to their lower occurrences in the corpus.

Table 3.15 also presents the frequencies of functional labels. SUBJ has the highest frequency in the corpus whereas OBJ label has less occurrences as compared to SUBJ. Both SUBJ and OBJ have been annotated along with NPs and PPs. When annotated with NPs, they mark nominal subject and object roles and with PP label, SUBJ marks ergative and dative subjects and OBJ marks accusative objects. There are many sentences without direct objects which resulted in low count of OBJ label. The labels OBL, ADJ, POS and PDL have quite covering frequency in the corpus. The labels VALA, VOC and INJ have lower frequencies as compared to other labels.

TABLE 3.16: Word statistics according to number of POS tags with number of unique words, their instances and coverage in the final treebank.

Category	No. of tokens	No. of instances	Percentage
Single POS words	13,148	83,722	56.3%
Two POS words	862	42,087	28.3%
Three POS words	88	14,793	10.0%
Four POS words	17	7,973	5.4%
Total	14,115	148,575	-

More than half of the word types have been consistently been annotated with one and the same POS tag as shown in Table 3.16. 28.3% of the word types were annotated with two different tags, 10% of the word types were annotated with three different tags and 5.4% of the word types were annotated with four different tags in the corpus. There are seventeen tokens which have four different POS tags, 88 tokens have been marked by three tags, 862 tokens used two tags and 13,148 token are tagged by using single POS tag in the corpus.

Table 3.17 shows domain-wise statistics of phrase labels. The number of sentences and tokens are different within text domains. Therefore, we have computed percentages of labels with respect to total number of labels. These statistics are helpful to understand syntactic and annotation variation across different domains. The labels NP, PP, VC, ADJP and ADVP have quite distributive frequency among the domains. The domain of letters has highest percentage of 'S' labels as compared to other domains. The same trend can be observed for SBAR label which also has highest percentage in Letters. In the annotation scheme, SBAR clause annotates subordinate clauses by annotating an inner clause with 'S' label which resulted in this similarity. In the same way, the lowest percentage of SBAR is 3.3% in press domain. The label 'S' also has lowest percentage for the same domain. The labels PREP, DMP and FFP have low frequencies in the corpus but they are present in several domains. Overall, the treebank has coverage of phrase labels across different domains.

Table 3.18 further shows domain-wise statistics of functional labels in the treebank. The percentages of labels are quite distributive among all domains. However, some domains have higher percentages. For example, NP-SUBJ has higher percentage for domains of entertainment, novels and science. Interestingly, domains of health and technology contain lower percentages of PP-SUBJ. These domains usually have texts in the form of instructions by using present tense and PP-SUBJ represents ergative and dative subjects which are usually annotated for past perfect sentences. Urdu uses complex predicate structure in all kind of texts which is shown by the coverage of POF label. POF label shows an even distribution in the corpus. Similarly, the copula construction marked with PDL, has representation in all domains. The label VOC and INJ have

TABLE 3.17: Domain-wise statistics (%) of phrase labels in the final treebank.

Domain	NP	PP	VC	ADJP	ADVP	S	SBAR	QP	PREP	DMP	FFP
Book reviews	37.5	21.5	17.1	3.2	1.4	14.2	3.9	1.0	0.02	0.08	0.14
Culture	34.8	20.8	18.0	3.9	1.7	15.1	4.9	0.7	0.02	0.0	0.04
Education	35.8	21.2	17.2	4.3	2.2	14.8	3.5	0.8	0.0	0.0	0.16
Entertainment	35.7	20.4	17.8	4.1	2.1	15.2	3.5	1.0	0.07	0.0	0.17
Health	32.3	18.7	19.1	4.4	2.3	17.5	4.3	1.2	0.04	0.04	0.09
Interviews	36.0	20.0	18.1	3.3	2.1	15.4	3.8	1.1	0.01	0.1	0.09
Letters	32.5	17.3	20.0	3.5	2.4	18.2	5.2	0.9	0.01	0.04	0.04
Novels	34.8	17.0	20.2	2.3	2.6	17.7	4.6	0.8	0.09	0.03	0.01
Press	37.1	21.9	16.8	3.7	1.8	14.2	3.3	1.2	0.02	0.04	0.05
Religion	37.1	18.9	18.8	2.8	2.1	14.9	4.6	0.7	0.03	0.03	0.06
Science	36.3	20.1	17.1	4.2	1.9	14.5	3.8	1.2	0.04	0.1	0.66
Short stories	35.5	16.2	20.1	2.7	2.9	16.9	4.8	0.7	0.03	0.07	0.02
Sports	35.7	21.3	17.3	3.8	2.0	14.3	3.6	1.8	0.0	0.05	0.17
Technology	36.2	19.5	17.7	4.2	1.9	14.3	3.8	1.1	0.0	0.29	0.93
Translations∗	36.3	17.5	19.6	2.6	2.2	16.5	4.2	0.8	0.03	0.06	0.04

∗Translations of foreign languages

lower percentages but they are part of several text domains. Conclusively, the treebank has coverage of all phrase and functional labels with respect to text domains as it has been developed by using a balanced corpus.

TABLE 3.18: Domain-wise statistics of functional labels in the treebank.

Domain	NP-SUBJ	PP-SUBJ	NP-OBJ	PP-OBJ	OBL	POF	PDL	ADJ	VOC	INJ
Book reviews	38.0	11.1	12.3	5.5	6.4	11.1	10.1	4.7	0.09	0.0
Culture	32.8	10.0	17.1	5.9	7.6	11.5	6.6	7.8	0.0	0.0
Education	37.1	9.0	14.9	3.5	4.9	13.8	9	6.1	0.0	0.14
Entertainment	40.8	6.8	12.9	2.9	5.5	10.8	9.5	9.8	0.14	0.0
Health	41.3	4.7	17.6	3.1	6.3	14.3	5.9	6.1	0.0	0.0
Interviews	38.3	9.3	15.4	2.3	5.6	12.9	7.6	7.2	0.29	0.06
Letters	38.5	5.5	20.8	4.0	5.9	11.7	7.6	4.8	0.25	0.13
Novels	41.2	11.0	15.2	2.6	3.6	10.8	4.8	8.5	0.89	0.14
Press	37.5	11.0	13.3	4.0	4.7	14.5	7.6	6.2	0.08	0.0
Religion	34.8	14.0	16.0	3.7	5.2	11.4	7.5	6.5	0.33	0.2
Science	41.9	8.0	12.1	3.6	5.7	11.1	9.1	6.8	0.18	0.0
Short stories	39.7	12.7	13.2	3.0	4.7	11.6	5.3	7.7	1.15	0.41
Sports	37.1	11.7	16.1	3.3	4.6	12.9	6.7	6	0.07	0.0
Technology	39.2	4.2	19.7	4.2	3.2	13.3	9.1	5.2	0.0	0.0
Translations∗	38.6	12.7	14.2	2.5	4.8	11.6	6.3	7.4	0.87	0.06

∗Translations of foreign languages

4. STATISTICAL PARSING

This chapter presents the parsing models and syntactic features which have been trained to parse the CLE-UTB. We discuss the preparation of data sets, statistical constituency parsers and syntactic features. We have performed experiments with several parsing formalisms which include probabilistic context-free grammars (PCFGs), lexicalized grammars, tree substitutions grammars (TSGs), recursive neural network based parsing and Bi-directional long-short term memory network based models. We have further performed transfer learning by training word representations to improve the parsing results.

Urdu is a morphologically rich language. Therefore, for statistical parsing, a large data set is required to cover all its surface forms. To overcome data sparsity, we have experimented with several linguistic and computational features to improve the statistical learning. These features include, POS tag upgradation, markovizations, Urdu-head word identification, lemmatization and word clustering. These features were helpful for the improvement of parsing scores when trained with different models. Unlexicalized PCFG based models and tree substitutions grammars have been trained by using discontinuous data-oriented parsing framework (disco-dop) [1]. Following sections describe our dataset, parsing models and features in more detail.

[1] https://discodop.readthedocs.io/en/latest/

4.1 DATASET

For the statistical parsing of morphologically-rich languages, the third issue is the data requirements as raised in [148]. A dataset with sufficient samples is crucial for training statistical parsers. The CLE-UTB uses a simplified annotation scheme for POS tagging as well as phrase labeling. Therefore, the treebank has sufficient samples to learn syntactic categories of the language. The corpus contains text from a number of domains written by several authors. Therefore, a balanced test set is necessary to present valid results. The test set has been prepared according to text domains as shown in Table 4.1.

The test set has been prepared by selecting random sentences with respect to number of sentences in different text domains. The test set also covers the various lengths of sentences. The length coverage of the test set is shown in Figure 4.1. The trend in the test set is quite similar to the overall length coverage of the whole corpus as shown in Figure 3.13.

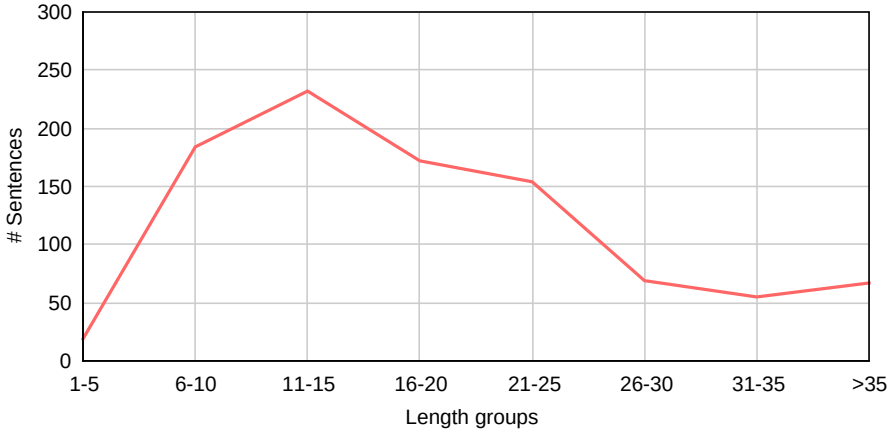


FIGURE 4.1: Length coverage of the test set.

It has been observed that parsers perform well on shorter sentences as compared

TABLE 4.1: Domain-wise train and test set division.

Sr#	Domains	# Train sents.	# Train tokens	# Test sents.	# Test tokens
1	Book reviews	320	7,399	70	1,277
2	Culture	369	7,087	52	1,044
3	Education	218	4,903	49	925
4	Entertainment	195	4,037	53	1,010
5	Health	482	8,614	52	980
6	Interviews	525	10,673	52	1,007
7	Letters	592	10,264	58	1,003
8	Novels	468	8,762	83	1,671
9	Press	372	8,743	52	1,036
10	Religion	438	8,462	50	892
11	Science	354	7,362	51	932
12	Short stories	1,496	24,589	168	2,974
13	Sports	454	8,980	51	953
14	Technology	85	1,691	39	773
15	Translations*	513	8,539	93	1,993
	Total	6,881	130,105	973	18,470

*Translations of foreign languages

to longer ones. Therefore, we have further divided our test set into three groups according to the number of tokens. These groups include, small sentences with 10 or less tokens, medium sentences with length between 11 and 25 and long sentences with 25 or more tokens. Table 4.2 shows the length based division of our test set.

TABLE 4.2: Length-wise division of the test set.

Test set	Small($\langle$=10)	Medium(11-25)	Long($\rangle$25)	Total
# Sentences.	229	526	218	973
# Tokens	1,919	9,136	7,415	18,470

The small set contains 229 sentences having 10 tokens or less, medium set contains 526 sentences with number of tokens between 11 and 25 and the long set contains 218 sentences with 25 tokens or more. The sub-categorization of the test set would be helpful for the analysis of the performances of the parsers. The annotation of the CLE-UTB covers all syntactic constructions and probable word orders of the language. However, an ordered test set has been prepared which contains sentences with different word orders. This test set has been used to analyze the parsers' learning for different word orders. Section 4.3.6 presents the ordered test set in detail.

4.2 CONSTITUENCY PARSERS

A constituency parser provides the analysis of a sentence and assigns the syntactic categories to the phrases and clauses. The syntactic structure is also helpful to understand semantics. The syntax is represented in the form of a parse tree. A parse tree shows the break down of a sentence into sub-phrases. A parse tree contains a root, non-leaf nodes and leaf nodes. In the annotation of the CLE-UTB, a root node has a label 'S', non-leaf nodes are annotated with phrase labels of the annotation scheme and leaf nodes are actual words of a language. The POS tags are referred as pre-leaf nodes as the parsers predict the phrase level relations whereas POS tags are predicted by using POS taggers. However, the accuracy of POS tagging is influential on the parsing scores.

The syntactic annotation of a language faces an issue of structural ambiguity. Generally, the ambiguity is referred as the possibility of multiple parse trees hence with multiple meanings of a single sentence. The structural ambiguity is quite common in

the natural languages. Figure 4.2 shows an example of structural ambiguity in an Urdu sentence.

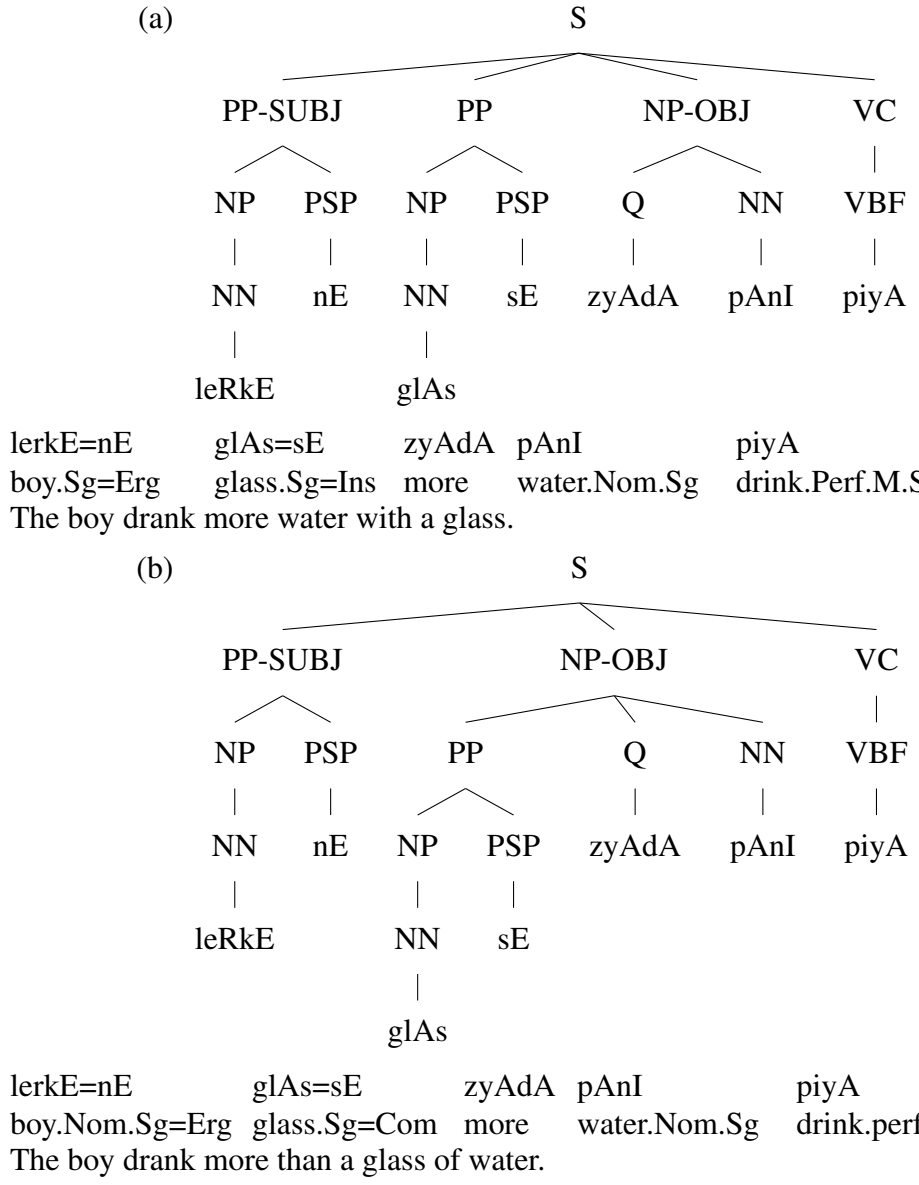


FIGURE 4.2: An example of structural ambiguity.

The example in Figure 4.2 shows two parse trees for an Urdu sentence. The post-positional subject and verb complex are similar in both parse trees. The attachment of the post-positional phrase changes the meaning in two representations. The first parse tree gives a meaning that the boy drank more water with a glass. The second tree depicts that the boy drank more than a glass of water. The grammar of a language may

generate more parse trees which shows severity of the structural ambiguity. Statistical parsers perform structural disambiguation and return most probable parse trees. In this work, we have experimented with a range of parsing techniques by using a number of syntactic features. Following sections describe grammar formalisms and parsing techniques which have been trained on the CLE-UTB.

4.2.1 Probabilistic Context-Free Grammars (PCFGs)

A PCFG is derived from an annotated corpus and is used for the prediction of parse trees for unseen sentences. The grammar rules are treated independent without using the contextual information. Each rule has a probability in the grammar. The probability is calculated by dividing the frequency of a rule by the frequency of the rule head. The left-hand side of a rule is referred as head of a rule. The head frequency is computed by counting the rules containing same head throughout the grammar of a treebank. Equation 4.1 shows the process to compute independent probabilities for grammar rules.

$$P(A \rightarrow B) = \frac{Count(A \rightarrow B)}{\sum_R Count(A \rightarrow R)} = \frac{Count(A \rightarrow B)}{Count(A)} \tag{4.1}$$

A is a single non-terminal and B is a sequence of terminal and non-terminal symbols. A is the head of the rule $A \rightarrow B$. The sum $\sum_R Count(A \rightarrow R)R$ presents the count of rules in which the head is A. The probability of a parse tree is computed by the product of probabilities of rules in the parse tree. The probability of a parse tree $P(T)$ can be calculated as shown by Equation 4.2.

$$P(T,S) = P(T) = \prod_{i}^{x} P(A \rightarrow B) \tag{4.2}$$

A is a single non-terminal and B is a sequence of terminal and non-terminal symbols. i is the starting grammar rule of a parse tree till the last x rule. The most probable parse tree can be computed by maximizing tree probabilities as depicted by Equation 4.3.

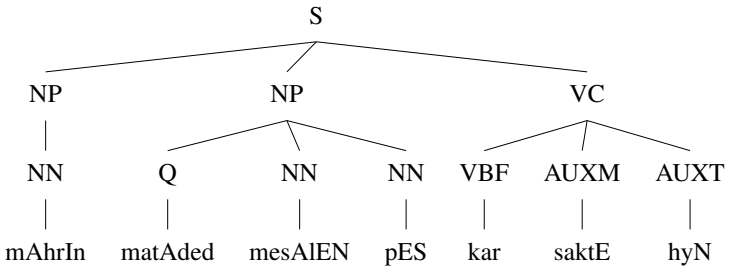
$$P(S) = \arg \max_{T \in parses(S)} P(T) \tag{4.3}$$

PCFGs make efficient parsers with respect to processing speed as they do not compute contextual information when implemented straight away. Context of phrases in a parse tree could be helpful to achieve more accurate parses. A few treebank representations could be helpful to embed contextual information with PCFGs. Section 4.3.2 describes the details of treebank representations for parsing with PCFG based parsers.

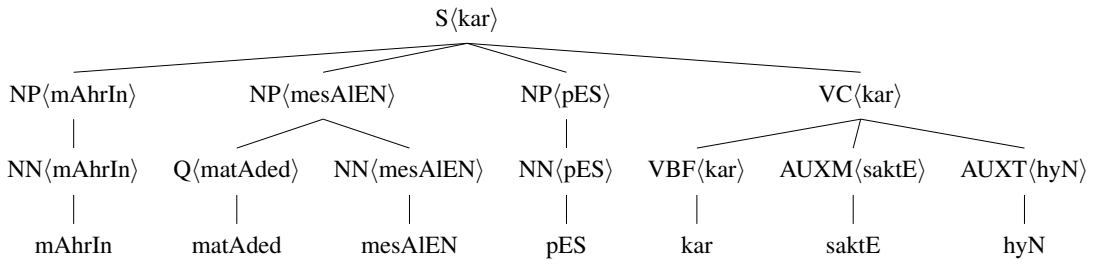
Lexical conditioning can provide the contextual information to the parse trees. The lexicalized grammars use the lexical information from phrases and sub-phrases. For this purpose, phrasal heads are computed to acquire the syntactic contribution of dominant words within the phrases. These grammars produced accurate parsers for English [38, 31, 104, 94]. The head word computation is a language dependent task. For parsing, an accurate head model is indispensable. Figure 4.3 shows an Urdu sentence parsed by using a simple PCFG and a lexicalized parser.

Both grammars have parsed the complex predicate structure into different phrases. A complex predicate structure is represented by noun *pES* 'to present' which is followed by a verb complex (VC). The head of the VC is a light verb *kar* 'do'. In the annotation

(a)



(b)



mAhrIn matAded mesAlEN pES kar SaktE hyN
expert.Pl many example.Pl present do.Sg can.mod.M.Pl be.pres.Pl
Experts can present many examples.

FIGURE 4.3: Two parse trees for the same sentence 'mAhrIn matAded mesAlEN pES kar SaktE hyN' (a) parse tree from a PCFG grammar (b) Parse tree via a lexicalized grammar.

guidelines, a complex predicate structure is annotated in two different constituents as shown by the second parse tree. The first parse tree is an output of a PCFG parser which did not include lexical conditioning hence produced an implausible structure. The example is evident that lexicalized parsers can produce better parsing results by employing a head-word algorithm. For experiments with lexicalized parsing, we have trained the parser presented in [94] by using an Urdu head model. Section 4.3.3 presents a head model for Urdu which we devised to perform parsing experiments on the CLE-UTB.

4.2.2 Tree Substitution Grammar (TSG)

TSGs make probabilistic parsers. They were proposed by [131] and were further formalized by [20]. The parsing based on TSGs is also called data-oriented parsing (DOP). A DOP model considers all possible subtrees in a parse tree and predicts parses for unseen sentences. These subtrees are called fragments. Several variations of DOP model are available with different applications. The DOP model can produce a huge number of fragments with respect to number of parse trees in a treebank. Many fragments may produce a lot of candidate parse trees for one sentence. The DOP model computes the probability distribution to perform disambiguation. It calculates fragment probabilities and derivations for estimation. The sum of probabilities of all fragments with the same root label is equal to one as shown by Equation 4.4.

$$\sum_{f \in Fx} P(f) = 1 \tag{4.4}$$

In Equation 4.4, Fx represents the set of all subtrees with root x. A sequence of fragments which produces a parse tree t by using the left-most substitution is called a derivation. Equation 4.5 shows the probability calculation for a derivation d.

$$P(d) = P(f_1, ..., f_n) = \prod_{i=1}^{n} P(f_i) \tag{4.5}$$

The relative frequency estimate (RFE) produces the simplest probability estimate against a fragment . RFE is calculated by dividing the frequency of a fragment f with sum of all fragments having same root in the treebank. $f' \in Froot(f)$ shows all

the fragments with same root.

$$P_{RFE}(f) = \frac{Count(f)}{\sum_{f' \in Froot(f)} Count(f')} \tag{4.6}$$

RFE does not provide a complete probabilistic interpretation. It contributes a biased estimate by assigning a higher probability value to larger fragments. Therefore, [66] proposed an equal weight estimate as shown by Equation 4.8.

$$W_{EWE}(f) = \sum_{t \in TB} \frac{Count(f,t)}{|\{f' \in t\}|} \tag{4.7}$$

$$P_{EWE}(f) = \frac{W_{EWE}(f)}{\sum_{f' \in Froot(t)} W_{EWE}(f')} \tag{4.8}$$

Equation 4.7 adds the division of sums against a fragment f for a tree t with all fragments. *P(f)* can be calculated by dividing the equal weight estimate by the sum of the equal weight estimates of all fragments having same root. To calculate the probability of a tree *t*, DOP sums the probabilities of all derivations of a tree *t* and maximizes the likelihood for the estimation of most probable tree as presented by Equations 4.9 and 4.10.

$$P(t) = \sum_{d \in D(t)} P(d) = \sum_{d \in D(t)} \prod_{f \in d} P(f) \tag{4.9}$$

$$t = argmaxP(t) \tag{4.10}$$

In this work, we have trained a double DOP model which was presented in [130]. The DOP model produced the contextual information by considering all possible subtrees. We have performed experiments for different subtree heights. The higher tree depths are helpful to attain more context of constituents.

4.2.3 Recursive Neural Network based Parser

A compositional vector grammar (CVG) was introduced by [141] which performs syntactic parsing by learning syntax and semantics of words in the treebank. They combined the syntactic categories achieved from a grammar based parser with a recursive neural network (RNN) model. Figure 4.4 shows a sample tree with a recursive neural network. The same weight matrix is replicated and used to compute all non-leaf node representations.

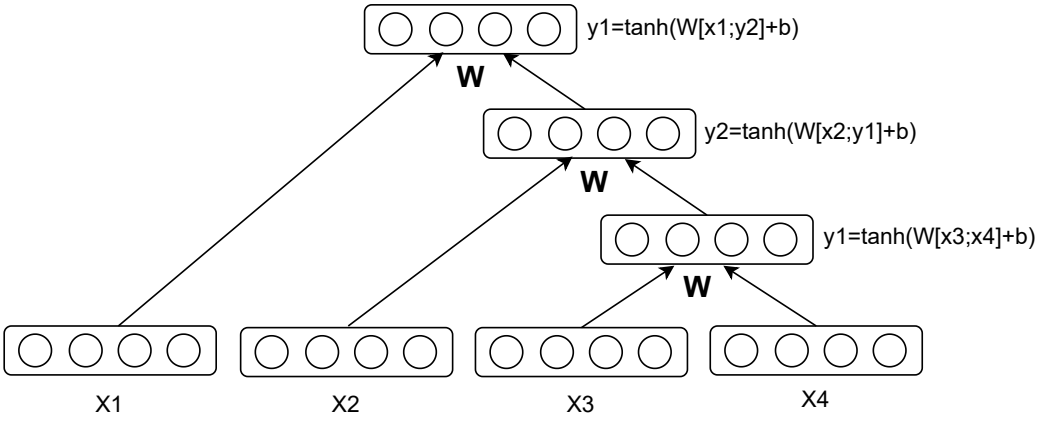


FIGURE 4.4: A sample tree with a simple recursive neural network.

The technique improved parsing results for English and achieved an f-score of 90.4. The parser was a continuation of parsing model proposed in [140]. Word embeddings were trained along with the parsing which were proposed in [149]. The POS tags were appended with the word vectors so that each pair contains a word representation

in the form of vector and POS tag. The CVG computes a score by adding up scores for all nodes of a tree. Max-margin algorithm performs predictions and objective function along with CVG and produces a tree with highest score as correct one from all possible parses. The objective function and optimization used for training are described by Equations 4.11, 4.12 and 4.13.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} r_i(\theta) + \frac{\lambda}{2}||\theta||_2^2 \tag{4.11}$$

$$r_i(\theta) = max_{\widehat{y}\in Y(x_i)}(s(CVG(x_i,\widehat{y})) + \Delta(y_i,\widehat{y})) - s(CVG(x_i,y_i)) \tag{4.12}$$

$$\Delta(y_i,\widehat{y}) = \sum_{d\in N(\widehat{y})} k\{d \notin N(y_i)\} \tag{4.13}$$

Where input sentence is represented as x_i, all possible parse trees for x_i are represented by $Y(x_i)$, correct prediction for x_i is y_i, $\Delta(y_i,\widehat{y})$ is the margin loss, k was set to 0.1 for all experiments, (x_i,y_i) represents all training samples, score function is represented by s, parameter collection is represented by θ, $\lambda = 10^{-4}$ for training and $N(y_i)$ represents all nodes for the parse tree y_i.

The word vectors have been used to find the correct parent for a pair of child nodes ($p \rightarrow ab$). A parent and its children have the same dimensionality. The vectors from child nodes are joined which result to have a dimension of $2n \times 1$. The dimension of weight matrix was set to $x \times 2n$ and *tanh* was used for nonlinearity. Similarly, the

output vector was used for the prediction of parent nodes. CVG considers the syntactic constructions extracted from the treebank by using a PCFG parser. The weights were learned by training a syntactically-united recursive neural network (SU-RNN). Child nodes were also influential in the weight matrix. A node score was computed by adding up the linear score of a parent and *log* probability of the rule ($P \rightarrow AB$) as demonstrated by Equation 4.14.

$$s\left(p^{(1)}\right) = \left(v^{(A,B)}\right)^T p^{(1)} + \log P\left(P_1 \rightarrow AB\right) \tag{4.14}$$

Where v is the parameter vector and the CVG computed scores for a parse tree by summing all node scores.

$$s\left(CVG\left(\theta, x, \widehat{y}\right)\right) = \sum_{d \in (\widehat{y})} s\left(p^d\right) \tag{4.15}$$

A lexicalized PCFG parser has been used for caching top 200 parse trees against each training sample of the CLE-UTB. RNN parser has been trained with an object as shown in Equation 4.11. Beam search was used to optimize the speed for top 200 parses. In the experiment, we set $\lambda = 10^{-4}$, the batch size was set to 20, the learning rate for AdaGrad was $\alpha = 0.1$ and word embedding dimensionality was set to 25. For our treebank, word embeddings were trained on a plain corpus having 35 millions Urdu words.

4.2.4 BiLSTM Parser (Proposed)

Recurrent neural networks (RNNs) are able to provide a learning framework to model word sequences by using the entire context. Our parsing model has been developed on bidirectional long-short term memory (BiLSTM) network, a variant of conventional RNN. For parsing, we have converted the problem to sequence labeling. The parse trees have been converted to linear structures along with tree labels and POS tags. The BiLSTM model was quite capable to predict the sequential labels. Figure 4.5 shows the model architecture having two LSTM layers, one in forward direction and other in backwards.

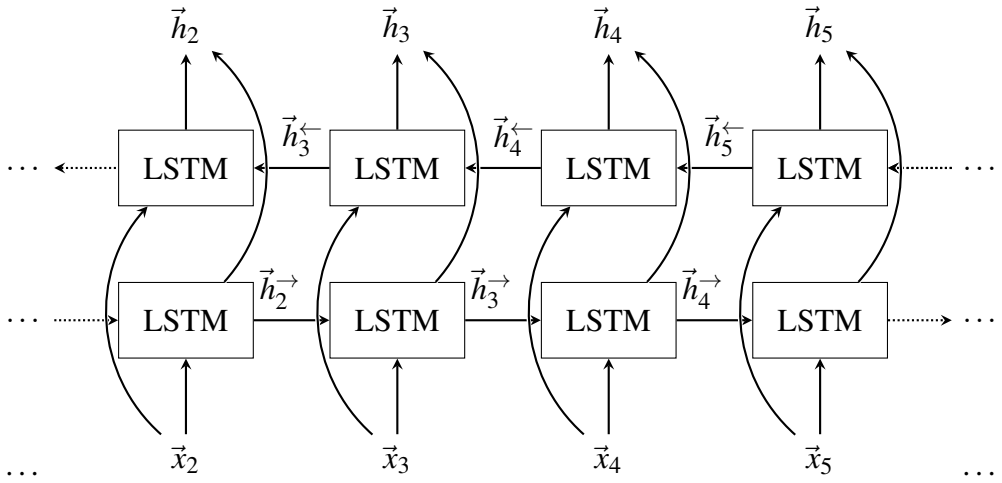


FIGURE 4.5: Bi-directional long-short term memory model for sequence labeling.

Conventional RNNs have the ability to learn shorter contexts but in case of longer sequences, they face a problem of vanishing gradient which makes them less practical for longer sequences. One solution is the LSTM networks which add the capability of forgetting the irrelevant previous information and remembers essential context. An LSTM cell contains multiple layers to perform tasks of forgetting and

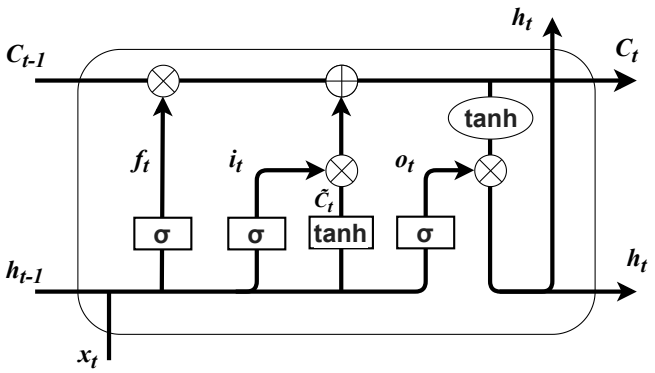


FIGURE 4.6: Basic structure of LSTM

remembering. Figure 4.6 shows the internal architecture of an LSTM cell. Equations 4.16, 4.17, 4.18, 4.19, 4.20 and 4.21 show the functions of hidden layers of a cell.

The first step of an LSTM cell decides what information to forget and what information to pass on to the next layer and to which degree. For that purpose, it takes x_t at time t and a previous hidden state $h_t - 1$ as input and outputs a number between zero and one by using a sigmoid (σ) layer also called forget gates and is represented by a function f_t (Equation 4.16). The number decided which percentage of the information to forget.

$$f_t = \sigma\left(W_f.[h_{t-1}, x_t] + b_f\right) \tag{4.16}$$

The second step decides about the storing of new information. This step works in two steps. These two layers are represented by functions i_t and $\tilde{C}_t$. i_t takes input x_t and the previous hidden state $h_t - 1$ and produces a number between zero and one. $\tilde{C}_t$ is computed in the form of a vector by using a *tanh* layer against input and the previous hidden state (Equation 4.17, 4.18).

$$i_t = \sigma\left(W_i.[h_{t-1}, x_t] + b_i\right) \tag{4.17}$$

$$\tilde{C}_t = tanh\left(W_C.[h_{t-1}, x_t] + b_C\right) \tag{4.18}$$

The third step updates the previous cell state C_{t-1} to produce new cell state C_t. For that purpose, forget layer f_t, information layer i_t and new candidate $\tilde{C}_t$ are used. At first, the previous cell state C_{t-1} is multiplied with f_t then add $i_t * \tilde{C}_t$ to achieve the new cell state C_t. The new cell state will be forwarded to the next LSTM cell $(t+1)$. The computation of cell state C_t is shown by Equation 4.19.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{4.19}$$

At the final step, the cell produces an output which is the hidden state of the cell. It is based on the new cell state. Similar to i_t, a sigmoid (σ) layer is used to decide what information to output (o_t) and then multiplies with the cell state after filtering it through a *tanh* layer. It is the output (h_t) of a cell which is forwarded to the next LSTM cell $t+1$. Equations 4.20 and 4.21 show the computations of o_t and h_t respectively.

$$o_t = \sigma\left(W_o.[h_{t-1}, x_t] + b_o\right) \tag{4.20}$$

$$h_t = o_t * tanh\left(C_t\right) \tag{4.21}$$

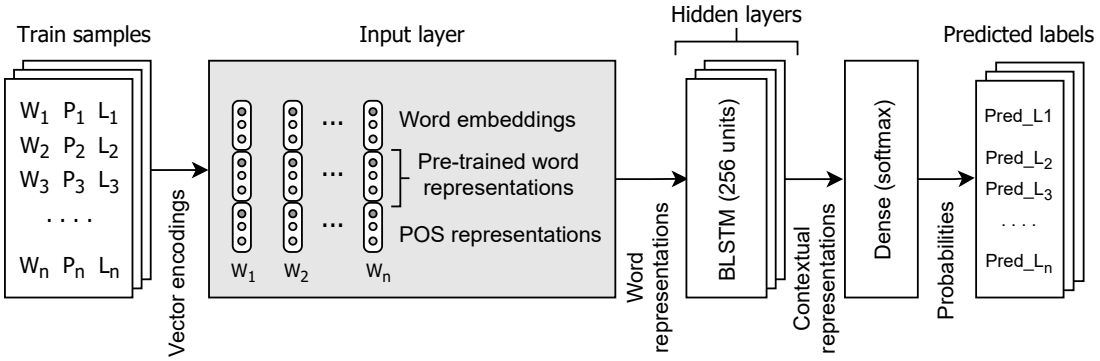


FIGURE 4.7: BiLSTM based parsing architecture.

LSTM based network architectures provide better learning for a sequence labeling problem. We have trained a bidirectional LSTM model to parse the Urdu treebank. The BiLSTM output vector at position i for the input sentence $w_{1:n}$ is defined by Equation 4.22. It is a vector h_i with conditioning previous context $w_1 : i$ and in addition, the next sequence $w_i : n$. A further dense layer was applied by using $softmax$ for multi-class classification which outputs the tree labels against the input sequence as shown by Equation 4.23.

$$BiLSTM(w_{1:n}, i) = LSTM_f(w_{1:i}) \circ LSTM_r(w_{n:i}) \tag{4.22}$$

$$o_i = Softmax(Wh_i + b) \tag{4.23}$$

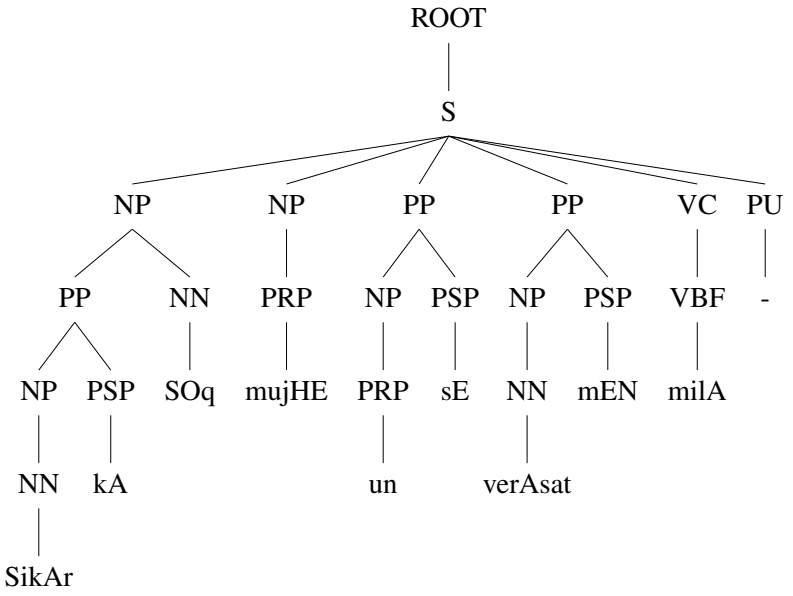
For a given sentence of n words $(x_1, x_2, ..., x_n)$, along with their POS tags $(t_1, t_2, ..., t_n)$ were represented by using embedding vectors. At index i, the input vector was computed by concatenating word vector $emb(w_i)$ and POS vector $emb(t_i)$ i.e. $emb(w_i) \circ emb(t_i)$. Figure 4.7 shows our long-short term memory networks based parsing architecture.

At first step, vector encodings are achieved from train samples. The input layer concatenates word embeddings, pre-trained word representations and POS embeddings to create a single vector for each token. These representations are further fed to hidden BiLSTM layers to achieve contextual representations. Finally, the output ($softmax$) layer performs the multi-class classification to produce the predicted labels. We further performed transfer learning to improve the learning of LSTM layers by including pre-trained word representations. The details of transfer learning are presented in Section 4.2.5.

4.2.4.1 Proposed Sequential Labeling

We transformed the parse trees into sequential format. For that purpose, we have used a baseline transforming method as discussed in [62]. The labeling output was in ConLL format in which token had POS tags and tree labels. The treebank labels contain a number and a label. The number which is the part of a label represents the number of common ancestors. The phrase label attached with the number which represents a common label between successive token at the lowest level in the tree. However, the unary branches were handled separately by attaching them with POS tags. Figure 4.8 shows an Urdu parse tree along with our proposed linear labeling as compared to relative labeling presented in [62].

The simple linear encoding produces a label by using the common number of phrase labels between w_i and $w_i + 1$. However, it usually produces large number of labels. The second encoding method proposed by [62] uses the difference of common



SikAr=kA SOq mujHE un=sE varAsat=mEN milA
hunting.Sg=Gen passion.Sg.Nom Pron.1.Sg pron.3.Pl=Abl inheritance.Sg=Loc find.Past.3.Sg
I inherited the passion of hunting from him.

SikAr	kA	SOq	mujHE	un	sE	varAsat	mEN	milA	-
Relative labeling of [62]:									
3_PP	-1_NP	-1_S	0_S	1_PP	-1_S	1_PP	-1_S	0_S	NONE
Absolute labeling:									
3_PP	2_NP	1_S	1_S	2_PP	1_S	2_PP	1_S	1_S	NONE
Proposed Labeling:									
PP	2_NP	1_S	1_S	PP	1_S	PP	1_S	1_S	NONE

FIGURE 4.8: A sample Urdu parse tree along with linearized labels.

ancestors from next $w_t + 1$ and previous token $w_t - 1$. For example, the token *kA* (genitive case marker) has a relative scale label -1_NP. The token has two common ancestors, NP and S, with next token *SOq* 'passion' and three common ancestors, PP, NP and S, with previous token *SikAr* 'hunting'. Therefore, the difference is -1 and the lowest common ancestor label is NP which produces a linear label -1_NP. This method is referred as relative scale and produces remarkably less number of labels as compared to absolute scale encoding. The unary branch labels were produced separately as first step of the

labeling which are further added with POS tags for conversion back to brackets. Therefore, experiments have been performed in two passes. First pass predicts unary labels and the second pass predicts the relative phrase labels. The output is further converted back to parse trees for f-score evaluation.

However, the analysis and experiments of relative and absolute sequential labeling has been performed for English Penn Treebank. The annotation of the CLE-UTB has relatively flat structure hence with less number of labels. Therefore, we have proposed a labeling scheme which has further fewer labels. The average entropy of our proposed labeling is also lower than the relative labeling. The most importantly, the proposed label set produces better parsing scores and outperformed the relative labeling up to one percent. Table 4.3 presents the comparison of both labeling schemes.

TABLE 4.3: Comparison of our proposed sequential labeling with the labeling of [62] with respect to the Urdu Treebank

Labeling	# of Labels	Average Entropy	Best F-score
Relative[62]	180	3.15	88.1
Proposed	143	2.65	89.1

Table 4.3 presents number of labels, average label entropy of all sentences and best parsing f-score when trained on the CLE-UTB. The entropy values depict randomness of the labels in a sentence. The entropy value of the proposed scheme is less than relative labeling showing that our labeling has less randomness hence have more learning potential in training. F-score values further represent the syntactic learning of our proposed sequential labeling. The proposed labeling scheme is based on the absolute labeling method which simply label common number of phrase labels of token and its

next token. The label further concatenates the lowest common phrase label to obtain a sequential label. A label contains a number followed by a phrase tag.

The absolute labels were analyzed against the annotation of the CLE-UTB to find predictable patterns. There were several patterns due to the simplified and flatter syntactic structure of the treebank. We have exploited them to reduce overall labels for training. For instance, the number of common phrases, for a PP label, is always one higher than the number of next label. In the absolute labeling, the first label is 3_PP and the second label is 2_NP. Similarly, fifth and seventh tokens have 2_PP label and their next label is 1_S. This sequence is found throughout the treebank. We therefore, dropped the number from all PP labels which reduced the size of label set. Same pattern was observed for PP-G labels and the labels were further reduced.

Another pattern was observed for the VC label. The number of common phrase labels are always one higher than the common phrase labels of the previous label in any sentence. The number was also removed from all VC labels in the treebank labeling. The demonstrative phrase (DMP) also showed a pattern which was similar to post-positional labels. The DMP labels were also simplified to reduce to labels by removing the number from them. These number were recovered back after get a predicted output of the parser. There were some patterns with NP labels as well but they rather decreased the parsing scores. However, the simplified labeling reduced the tag set to 143 as compared to relative labels and produced an improved parsing f-score of 89.1 as shown in Table 4.3.

BiLSTM models can process inputs of arbitrary length. In case of batch processing, the longest sentence of the batch determines the length and shorter sentences

are padded to this length. Therefore, we limited the model to use 100 tokens for each sentence. The embeddings of words and POS tags were concatenated to achieve input vectors. Two hidden LSTM layers have been used having 200 dimensions each. The Adam optimizer was used with a learning rate of 10e-3. The training was done for 20 epochs with a batch size of 64. We further developed a POS tagger which was also based on LSTM networks. The tagger was trained with a single bi-directional LSTM layer having 256 dimensions and trained for 24 epochs. For BiLSTM parser and POS tagger, *keras* open-source software library has been used with *tensorflow* background. The performance of both, parser and tagger, was improved by employing transfer learning.

4.2.5 Transfer Learning

The transfer learning uses the solution or knowledge from one problem and applies it to solve another but related problem. For instance, annotated data for tasks of POS tagging and parsing may not cover all possible constructions for a language because the process of annotation is quite costly. In this case, transfer learning becomes a suitable option. Pre-trained models are usually used to embed into the current models. We have performed syntactic parsing and POS tagging by using transfer learning. Word representations have been trained on large plain Urdu corpus. We have experimented with Word2Vec [110] and deep contextualized word representations (ELMo) [123] for parsing and POS tagging by using BiLSTM models. For parsing, the same corpus was used to train word representations with RNN and BiLSTM parser. The POS tagger achieved better accuracy with ELMo embeddings while the parser performed well with

Word2Vec embeddings.

To train word representations, the Urdu corpus contained 1.71 million sentences with about 35 million words[2][3].Word2Vec embeddings were trained with a vocabulary size of 72 thousands and 100 dimensions for each vector. However the ELMo representations were trained for 128 dimensions. The parsing results by using transfer learning are presented in Chapter 5.

4.3 SYNTACTIC FEATURES

To parse morphologically rich languages, the language representation, input type, label sets and the size of training set are important. The models for constituency parsing may not work well on morphologically rich languages like Urdu. Therefore, it is important to use syntactic features in the training process for the achievement of better results. The features should be selected in such a way that they are helpful to overcome the issue of data sparsity and provide better syntactic learning. We have investigated a number of features which were helpful to improve parsing scores. A simple PCFG model has been used as baseline parser in our experiments and it uses default language representation and POS tag set. Following sub-sections describe syntactic features in more detail.

4.3.1 Updated POS Tags (POS2)

The existing POS tag set is designed to have fewer tags resulting to be flat for several word classes. The tag set has been designed to cover the morphologically rich

[2]http://www.cle.org.pk
[3]https://www.urdudigest.pk

nature of the language. The post-positional tag, for example, has a single tag PSP, regardless of syntactic usage of the clitics. A single tag has been used to mark all types of common nouns. In the same way, single tag for adjectives, adverbs etc. The punctuation symbols are also tagged by using a single tag, PU. However, the well-known Penn Treebank POS tag set has several tags for different types of prepositions and punctuations. Therefore, we have experimented with different syntactic sub-categorization of the tag set and proposed an updated version based on the initial parsing results. We have divided the tags for post-positions and punctuation into additional categories as shown in Table 4.4.

TABLE 4.4: Updated POS tags for post-positions and punctuations.

Category	Clitic	POS	Updated POS	Description
Post-position	*nE*	PSP	PSP-E	Ergative
	kO	PSP	PSP-A	Accusative/Dative
	kA, *kI*, *kE*	PSP	PSP-G	Genitive
	sE	PSP	PSP-SE	Ablative/Instrumental
	az, *tA*	PSP	PSP-I	Internal
	-	PSP	PSP	All other post-positions
Punctuation	‘،’,‘؛’	PU	PU-C	Comma/Semicolon
	‘۔’,‘؟’	PU	PU-P	Period/Question mark
	‘ ! ’,‘ : ’	PU	PU-E	Exclamation/Colon
	-	PU	PU	All other punctuations

The categorization of cases and case markers has been described in [85]. The tag PSP-E represents the ergative case which uses a case marker *nE*. It appears to mark an ergative subject of a sentence. The tag PSP-A has been categorized to mark accusative and dative case markers. Both cases use the same case marker *kO*. Similarly, three clitics (*kA*, *kI* and *kE*) represent genitive cases and we have given them a single tag PSP-G.

Ablative and instrumental cases are represented by a case marker *sE* with an updated tag PSP-SE. Urdu also uses internal clitics like *az* and *tA*, for example, *kam az kam* 'at least'. The POS tag set annotates such clitics as post-positions. We have categorized such clitics to have an updated tag PSP-I because their syntactic structure is different from other post-positions. All other case markers were tagged with the tag PSP.

In the same way, the tags for punctuation symbols have been divided into four categories. Commas and semicolons have been tagged with PU-C, periods and question marks have been marked with PU-P, the exclamation symbol and colon are marked with PU-E and all remaining punctuations have been tagged by using the default PU tag. The tagging division has been performed on the bases of initial parsing experiments. The sub-categorization of these POS tags helped to achieve better parsing results by keeping its size to be practical.

4.3.2 Markovization

Context-free grammars induce the grammar rules from a treebank by considering each rule independent of other rules. The probabilities are calculated by counting the occurrences of specific constituents and their rules heads (left-hand sides). For example, production rules with head NP are treated and computed in a similar way even if they have different parents. Figure 4.9 shows a parse tree with two NPs.

The NP attachment under a PP represents the subject of the sentence as it is

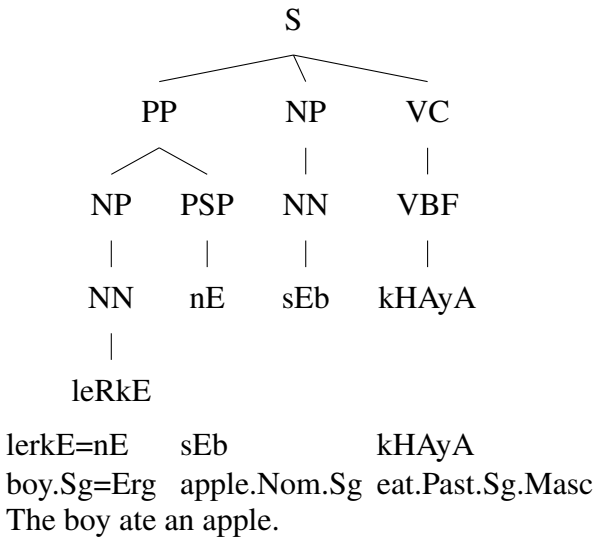


FIGURE 4.9: A parse tree for a sentence 'leRkE nE sEb KHAyA' without contextual annotation.

followed by an ergative case marker *nE*. On the other hand, the second NP has been attached under 'S' label and it represents the object of the sentence. The example demonstrates that the independent rule probability may not produce complete syntactic information about some constitutes. However, a treebank representation method is available which could help to include contextual information in the parse trees. A parental annotation method was proposed in [80]. Figure 4.9 shows the updated parse tree by using parental annotations.

The parental annotation attaches the parent label with the child node which results into a new label. The node annotation can be vertical as well as horizontal. In the horizontal annotation, the sibling nodes are attached with tree nodes at the same depth. The node annotation method could be helpful in parsing when treebank annotation is flat. However, too much higher parental or sibling annotation depth may result into data sparsity. To achieve the optimal annotation depths, a series of experiments have

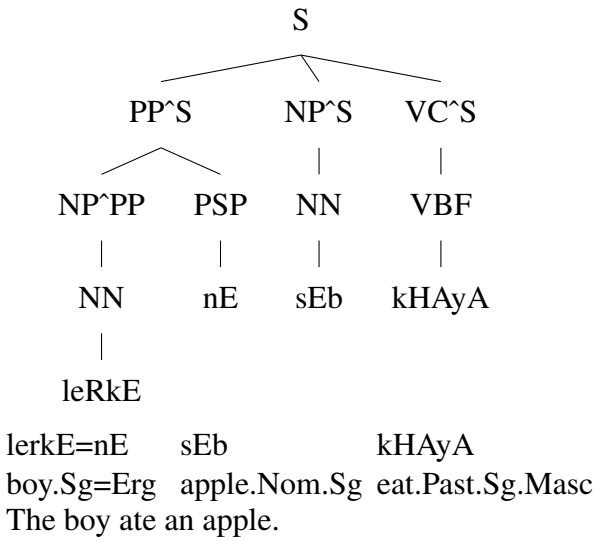


FIGURE 4.10: Parse tree for the sentence in Figure 4.9 with parental annotation.

been performed on the Urdu treebank. We have experimented with both horizontal and vertical markovization values. Figure 4.11 shows f-scores based on different values of vertical and horizontal markovizations. It is clear that by increasing the vertical and horizontal contexts, the parsing scores would be improved. However there is a subtle difference in the results for values between two and three.

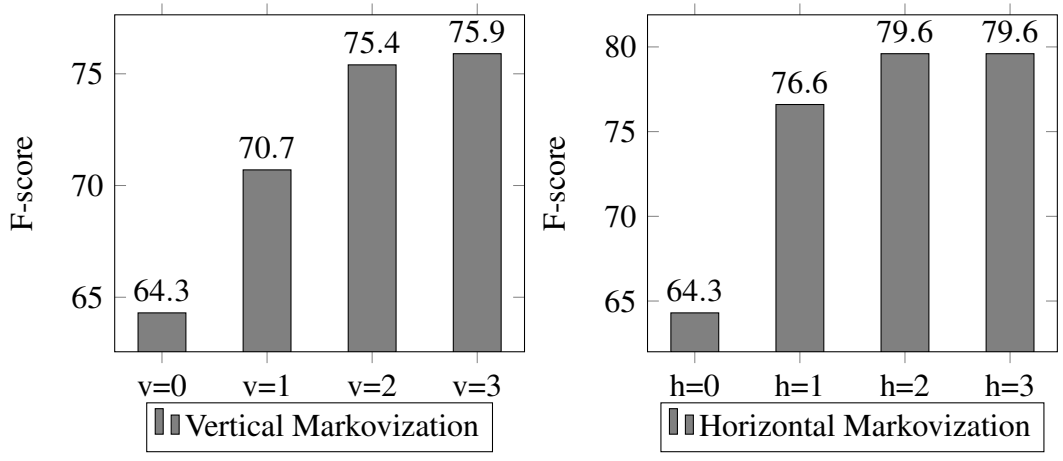


FIGURE 4.11: The parsing scores with respect to vertical and horizontal makovizations.

We have further experimented with different combinations of markovization values. Table 4.5 shows a confusion matrix for both markovizations when set from zero to

three.

TABLE 4.5: F-scores with respect to vertical and horizontal makovization values when evaluated on the test set by training a PCFG parser.

Markovization	h = 0	h = 1	h = 2	h = 3
v = 0	64.3	76.6	79.6	79.6
v = 1	70.7	78.5	79.6	79.9
v = 2	75.4	80.1	80.6	80.5
v = 3	75.9	80.7	81.5	80.8

It is important to select optimal markovization values during experiments as higher values could create an issue of data sparsity by increasing the size of the label set by appending the contextual labels. For our parsing experiments we have selected the value of two for both vertical and horizontal markovizations. All grammar-based parsing models have been trained by setting these values.

4.3.3 Head-Word Model

Lexicalized grammars use phrasal heads to learn the contextual information. The head words provide syntactic contribution of phrases in a sentence. Head-driven grammars produced more accurate parsing models for English [36, 38, 94]. Simple PCFG based parsers do not consider lexical conditioning in the parse trees. For parsing of morphologically rich languages, like Urdu, could be a challenging task as it has many surface forms. However, the lexical conditioning could be used to reduce ambiguities [38]. To implement lexical conditioning in a grammar based parser, an accurate head-word model is indispensable. We have devised a model for Urdu and evaluated it on a

small head-based test set. This test set contained 100 sentences which were annotated by phrasal heads manually.

4.3.3.1 Head-Word Algorithm

The extraction of head-words is a language specific task. One way is a straight forward left head identification which will make the left most word as the head of a phrase. This kind of head model would work for languages having fixed word order. For Urdu, we need to devise a model which is appropriate for the language. We have devised a head model for Urdu which is based on the syntactic annotation of phrases. Urdu is a head-final language and has a right most words as phrasal heads in most of the cases. The verbal structure of Urdu contains a main verb followed by auxiliary verbs making first lexical item as head. Syntactic cases are usually represented by using case markers which appear after an NP. Therefore, PPs will have left head like verb complex (VC) phrase. For other phrases including NPs, ADJPs, QPs, ADVPs, PREPs and DMPs, we have right most words as heads. The phrases, SBAR and FFP have left heads. The head on an S is the first verb from left to right. The POS tags and phrase labels further help to find accurate heads. Table 4.6 shows the mechanism of our head-word model.

The left most column in Table 4.6 shows phrase labels. The second column presents the search direction to find a head-word based on the POS tags and phrase labels given in third column. The priority of tags and labels is from left to right as shown in the third column. For a label, the algorithms finds the head according to the given direction either left to right or right to left from the list of tags or labels given in the third column. However, the list of tags is scanned from left to right. A noun phrase

TABLE 4.6: Urdu head-word identification.

Phrase labels	Direction	Tags/Labels
VC	left to right	VBF, VBI, AUXA, AUXT, AUXM, AUXP, VC, ADJP, NN, NP
PP	left to right	NN, NNP, NP, PSP-G, PSP-SE, PSP-NE, PSP-KO, PSP-I, PSP
NP	right to left	NN, NNP, ADJP, CD, JJ, RB, QP
	left to right	NP
ADJP	right to left	JJ, QP, NN, ADVP, VBI, AUXP, ADJP, NP, Q, SBAR, RB
QP	right to left	NN, JJ, RB, Q, CD, OD, QP
ADVP	right to left	RB, ADVP, CD, JJ, SCK, NP, NN
S	left to right	VC, S, SBAR, ADJP, NP
SBAR	left to right	SCK, Q, S, SBAR
PREP	right to left	NP, NN, NNP, PRE
DMP	right to left	PDM, PRP
FFP	left to right	FF

has two possibilities hence to search head in both directions. It has a right head with respect to given labels but we chose left word if an NP is annotated to have more than one NPs in it. The head word model has been used for training grammar based parsing models.

4.3.3.2 Urdu Head Model

The proposed model has been evaluated on manually annotated 100 sentences. The sentences were annotated independent of the head model based on the content words according to the context. The general language intuition may not follow the devised model for all cases. Table 4.7 presents results of our proposed head model.

TABLE 4.7: Head model evaluation for manually annotated sentences.

Constituent	Count	Accuracy
NP	631	96.3%
PP	274	100%
VC	224	99.5%
ADJP	36	91.7%
ADVP	25	96.0%
S	261	86.6%
SBAR	66	90.9%
QP	16	87.5%
Overall	1,536	95.4%

For the evaluation, random sentences were selected which contained more frequent phrase labels as shown in Table 4.7. It presents results for eight phrase labels. Other labels including PREP, DMP and FFP have less samples as shown in Section 3.5. The table also shows the frequency of phrase labels. The highest accuracy was achieved for PP phrases. VC phrases have an accuracy of 99.5%. Some phrases like S and QP produced lower accuracy. However, The overall accuracy for automatic phrasal head identification is above 95%.

4.3.4 Lemmatization

An Urdu verb has seventeen surface forms of non-causatives verbs. It has same number of additional forms for causatives and bi-causatives [79]. Urdu verbs have further underlying forms for four honorifics. For nouns, it has surface forms for number, gender and case. A sufficient data set is required to train statistical parsers for a morphologically-rich language. Therefore, the lemmatization process is anticipated to

be helpful to reduce the data sparsity. It will replace the surface forms with root forms. We have trained lexicalized parser after performing lemmatization on the train and test sets. Lemmas have been mapped for open class words based on their POS tags. An existing finite state morphological analyzer has been used for lemmatization as described in [79]. The process was carried out for verbs, common nouns, adverbs and adjectives. We have performed parsing experiments in three steps. In the first step, all the fours classes were replaced with root forms. In the second step, nouns, adjectives and adverbs were replaced with root forms and in the third step, only verbs were mapped on their roots. All verb classes were used in the lemmatization which include; infinite verbs, finite verbs and all four auxiliary verbs.

4.3.5 Word Clustering

Lemmatization reduces the number of inflectional forms in the corpus. However, the parsing accuracy is dependent on the coverage and performance of lemmatizers. To cope with the data sparsity issue, we further adopted a word clustering mechanism. A clustering algorithm creates groups of words based on syntactic similarities. Each cluster is labeled with a unique tag. This process helps to group open class words into different clusters according to their syntactic usage in the corpus. The categorization is dependent on the chosen number of clusters. For this purpose, we trained a predictive exchange word clustering algorithm[4] presented in [48]. The clustering is based on a predictive exchange algorithm [67] and is called bi-directional, interpolated, refining and alternating (BIRA) algorithm. The algorithm performs clustering on bi-directional

[4]https://github.com/jonsafari/clustercat

bigram language models with alternating interpolated weights for iterations. The clustering process is shown by Equations 4.24 and 4.25.

$$P(w_i|w_{i-1},w_{i+1}) = P(w_i|c_i)\,(\lambda P(c_i|w_{i-1}) + (1-\lambda)\,P(c_i|w_{i+1})) \tag{4.24}$$

Where w_i refers the ith word, c_i is the ith class and λ is the interpolated weight.

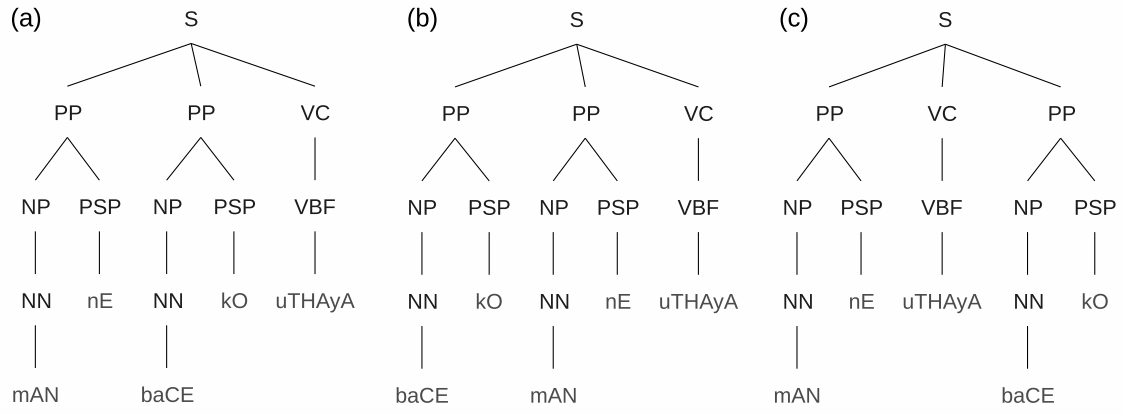
$$\lambda_i = \begin{cases} 1-\lambda_0 & if \quad i \bmod a = 0 \\ \lambda_0 & otherwise \end{cases} \tag{4.25}$$

The weight λ is equal to $1-\lambda$ when $i \bmod a = 0$ and a is the number of the iteration. The running time of the algorithm is $O(2\times(B+|V|)\times|C|\times I)$. Where B is the total number of bi-grams, $|V|$ is the corpus vocabulary, $|C|$ is the number of clusters to be computed and I is the number of iterations performed for clustering. A plain text corpus has been used to achieve cluster classes. We achieved seven clustered datasets based on the number of clusters in them ranging from one thousand to seven thousand clusters. The lexicalized parser has been used for each cluster labeling and results were analyzed.

4.3.6 Free Word-order

Urdu has a flexible word order like many other South Asian languages [126]. The property of the language allows different word orders of a sentence bearing plausible syntax with the same meaning. A treebank should cover probable word orders of

a language. A suitable parsing method is also essential to learn this property of a language. In Urdu, the phrases can have a flexible order but within phrases the word order remains intact mostly. Different word orders of a sentence are shown by Figure 4.12.



(a) mAN=nE baC-E=kO utH-A-ya
mother.SG=ERG child.SG=ACC pick.CAUS.PAST.SG.MASC
Mother picked up the child

(b) baC-E=kO mAN=nE utH-A-ya
child.SG=ACC mother.SG=ERG pick.CAUS.PAST.SG.MASC
Mother picked up the child

(c) mAN=nE utH-A-ya baC-E=kO
mother.SG=ERG pick.CAUS.PAST.SG.MASC child.SG=ACC
Mother picked up the child

FIGURE 4.12: Parse trees with three word orders of the same sentence 'mAN nE baCE kO utHAyA'.

Figure 4.12 shows three most probable word orders in our corpus. However, more combinations of phrase orders can be produced but we chose three word orders according to their frequent occurrences in the corpus. These word orders include SOV (subject-object-verb), OSV (object-subject-verb) and SVO (subject-verb-object). To analyze the learning ability of parsers on different ordered sentences, we prepared an ordered test set which contains sentences of these word orders. The sentences were selected from the existing test corpus having length of fifteen tokens or less. Subordinate

and coordinates conjunctions were avoided to create a test set with unambiguous orders. Table 4.8 shows the details of our ordered test set.

TABLE 4.8: Ordered test set having three word orders.

Word orders	SOV	OSV	SV	All
# Sentences	83	80	147	310
# Tokens	1,015	922	1,207	3,144

The ordered test set contains sentences of SOV, OSV and SV structures. The OSV structure has less frequency in the corpus. Therefore, we updated some of the sentences to have OSV order from SOV. For this purpose, subject and object positions were swapped. The parsing results were computed against gold and predicted POS tags. Chapter 5 presents the parsing results against ordered test set and their interpretations.

5. RESULTS AND DISCUSSIONS

We have parsed the treebank by training the parsing techniques presented in Chapter 4. These techniques include grammar-based parsers, tree substitution grammars, recursive neural networks based model and the proposed bidirectional long-short term memory based parser. Non-lexicalized and lexicalized grammars have been trained under grammar-based models. Beside the parsing models, several syntactic features have been trained to improve the parsing results. Updated POS tag set and markovizations were experimented with all grammar based models. Lemmatization and word clusters were further used with lexicalized parser. The ordered test set was evaluated for data-oriented parser. The word representations were trained to perform transfer learning with RNN and BiLSTM parsers.

5.1 CONSTITUENCY PARSING RESULTS

We have used Parseval measures to calculate parsing results. The measures compute labeled recall, labeled precision and labeled f-scores. The labeled measures consider a constituent correct if the constituent is identical in the reference corpus with respect to phrase labels and contains same tokens having correct phrasal boundaries. Equations for Parseval measures (Recall, Precision and F-score) are given in Section 2.3.4.

The labeled f-score was computed by giving equal weights to recall and precision. Therefore, f-score is referred as F_1. The test set was further divided into three subsets containing sentences of categories; small, medium and long. Additionally, we have evaluated the parsers against gold as well as predicted POS tags. For the gold POS tags, we have used original text along with actual POS tags. The predicted tags were achieved by using a POS tagger. The tagging performance was achieved as accuracy to show the percentage of correct tags predicted with respect to gold tags. Following sections present the parsing results and discuss their interpretations.

5.1.1 Grammar-based models

This section presents the results for constituency parsing achieved by training the grammar-based parsers. They include simple PCFGs[1] and lexicalized PCFG [94] parsers. The experiments have been performed by using gold POS tags as well as predicted tags. Table 5.1 presents the parsing results.

Table 5.1 further shows the results for the same parser by using predicted POS tags. It presents baseline f-scores for our treebank by training PCFG parser which are further trained by using different features. The table shows columns for parsing models, evaluation measures used, scores against small, medium, long sentences and accumulative scores for the whole test set. The accumulative f-scores have been highlighted with bold text. It is quite clear that parsing evaluations by using gold POS tags are higher as compared to predicted tags. Results further show that the syntactic features are helpful to improve parsing scores. Starting from our baseline PCFG f-scores which are 76.2 and 71.4 for gold and predicted POS tags, the f-scores were improved significantly by using

[1] https://discodop.readthedocs.io

TABLE 5.1: Grammar-based parsing results by using gold and predicted POS tags.

Parsing models	Urdu text + Gold POS				
	Measure	Small	Medium	Long	Accum.
Baseline PCFG	*LR*	86.6	78.0	70.4	75.8
	LP	85.7	78.3	72.3	76.7
	F_1	86.1	78.1	71.3	**76.2**
PCFG + POS2	*LR*	90.7	84.5	78.5	82.7
	LP	89.2	84.8	80.5	83.6
	F_1	90.0	84.7	79.5	**83.1**
PCFG + POS2	*LR*	91.2	85.3	82.7	84.9
+Markovization	*LP*	89.7	84.9	83.7	84.9
	F_1	90.5	85.1	83.2	**84.9**
Lex-PCFG	*LR*	92.5	85.5	80.9	84.3
+Left-Head	*LP*	91.0	83.4	80.0	82.8
	F_1	91.7	84.4	80.4	**83.5**
Lex-PCFG	*LR*	92.7	88.0	84.6	87.1
+Urdu-Head	*LP*	89.8	85.7	83.0	85.1
	F_1	91.2	86.8	83.8	**86.1**
	Urdu text + Predicted POS				
	Measure	Small	Medium	Long	Accum.	POS
Baseline PCFG	*LR*	80.2	72.9	65.4	70.6	94.5%
	LP	80.3	73.8	68.0	72.2	
	F_1	80.3	73.4	66.7	**71.4**	
PCFG + POS2	*LR*	84.7	80.0	74.2	78.2	94.9%
	LP	84.3	81.0	76.9	79.7	
	F_1	84.5	80.5	75.5	**78.9**	
PCFG + POS2	*LR*	85.3	80.9	77.7	80.1	95.1%
+Markovization	*LP*	84.9	81.4	79.4	81.0	
	F_1	85.1	81.2	78.5	**80.5**	
Lex-PCFG	*LR*	88.6	82.3	77.7	81.1	95.2%
+Left-Head	*LP*	87.3	80.2	77.2	79.7	
	F_1	88.0	81.3	77.5	**80.4**	
Lex-PCFG	*LR*	87.7	85.4	81.8	84.2	95.7%
+Urdu-Head	*LP*	85.0	83.1	80.3	82.2	
	F_1	86.3	84.2	81.0	**83.2**	

updated POS (POS2) tags as presented in Section 4.3.1. The scores were improved to 83.1 and 78.9 for both test sets. Vertical and horizontal markovizations were further experimented along with POS2 tag set. The f-scores were improved to 84.9 and 80.5

for both input representations. These features were also helpful to improve the POS tagging accuracies.

Table 5.1 also presents experiments for the lexicalized PCFG parser by using two head-models. However, for the experiments, we used the same setup containing POS2 and markovization values. The left-head model uses the left factored grammar by employing left-hand-side word as head for all constituents. The results are 83.5 and 80.4 for gold and predicted tags which are even lower than baseline PCFG. These results are evident that Urdu does not follow a left-head model and it requires a sophisticated, language dependent head finding mechanism. Therefore, we then experimented by using the proposed head-model which has been described in Section 4.3.3. The Urdu head-model provided improved results with f-scores of 86.1 and 83.2 for gold and predicted tags and the POS tagging accuracy was also improved to 95.7% which is quite promising for a morphologically-rich language Urdu. The morphologically-rich nature of the language could cause data sparsity for statistical parsing. Therefore, we experimented with lemmatization by training the same lexicalized parser. The results are presented in Table 5.2.

We implemented a lemmatizer by using an Urdu morphological analyzer [79]. The experiments have been done in three steps. The first step replaces the lemmas for common nouns, adjectives, adverbs and all types of verbs which are part of a verb complex. We achieved parsing f-scores of 85.9 and 82.4 for gold and predicted POS tags when trained the lexicalized parser. However, these scores are lower when trained without lemmatization. In the second step, we used lemmas for nouns, adjectives and adverbs. The scores against gold POS remains same to 85.9 and there was an increase

TABLE 5.2: Lexicalized parsing results with lemmatization.

Parsing models		Urdu text + Gold POS				
	Measure	Small	Medium	Long	Accum.	
LexPCFG + Lem.	*LR*	93.0	87.8	84.4	87.0	
(NN/JJ/RB/VC)	*LP*	90.4	85.6	82.7	85.0	
	F_1	91.7	86.7	83.6	**85.9**	
LexPCFG + Lem.	*LR*	92.7	87.8	84.3	86.9	
(NN/JJ/RB)	*LP*	89.8	85.5	82.7	84.8	
	F_1	91.2	86.6	83.5	**85.9**	
LexPCFG + Lem.	*LR*	93.1	88.1	84.6	87.2	
(VC)	*LP*	90.5	86.0	82.9	85.2	
	F_1	91.8	87.0	83.8	**86.2**	
		Urdu text + Predicted POS				
	Measure	Small	Medium	Long	Accum.	POS
LexPCFG + Lem.	*LR*	87.4	84.5	80.9	83.4	94.5%
(NN/JJ/RB/VC)	*LP*	85.1	82.3	79.4	81.4	
	F_1	86.2	83.4	80.2	**82.4**	
LexPCFG + Lem.	*LR*	87.5	85.1	81.6	83.9	95.5%
(NN/JJ/RB)	*LP*	85.1	82.8	79.9	81.9	
	F_1	86.3	83.9	80.7	**82.9**	
LexPCFG + Lem.	*LR*	88.3	85.4	82.3	84.4	95.1%
(VC)	*LP*	85.7	83.1	80.7	82.4	
	F_1	87.0	84.2	81.5	**83.4**	

of 0.5 in the f-score for predicted tags. The POS accuracy also increased from 94.5% to 95.1%. The third step used the lemmas for only verb complex by replacing lemmas for non-finite and finite verbs and all types of auxiliary verbs. The achieved f-scores are 86.2 and 83.4 which are a little higher then the scores without lemmatization. We evaluated the morphological analyzer on quantitative bases to analyze the contribution of lemmas in parsing. Table 5.3 shows the quantitative results of the morphological analyzer on the treebank.

TABLE 5.3: Quantitative evaluation of the lemmatizer.

Category	Verbs	Auxiliary verbs	Nouns	Adjectives	Adverbs
Percentage	69.5%	74.7%	20%	18.1%	10.9%

Table 5.3 shows the percentage of different categories which were mapped on their lemmas. Under verb complex, 69.5% verbs and 74.7% of auxiliary verbs were mapped on their lemmas in the corpus. On the other hand, the percentage of nouns, adjectives and adverbs were 20%, 18.1% and 10.9% respectively. The percentage of verbs is higher and hence produced relatively higher f-scores. It can be concluded that lemmatization helps to improve lexicalized parsing. However, the accuracy of the lemmatizer is largely influential on the results.

We adopted another method to cope with the data sparsity of the language by using word clusters. For that purpose, an unsupervised clustering algorithm was trained to achieve the word clusters. Words were grouped into clusters having similar syntactic contributions in the corpus as presented in Section 4.3.5. We performed parsing by replacing words with their cluster labels. We performed five experiments with lexicalized parser starting with one thousand clusters up to seven thousand clusters. Table 5.4 presents parsing results against different number of clusters. The experiments also have been performed by including gold and predicted POS tags.

The word clusters provided improved f-scores as compared to lemmatized corpus. Even with one thousand clusters, the parsing scores were higher than the lemmatized parsing. As we increase the number of clusters, the parsing scores show an upwards trend till five thousands clusters by using the gold tags. The f-score against five thousand clusters is 86.9. However, the f-scores for small test set produced better results when trained with one thousand clusters as compared to five thousands. Similar trend can be observed for predicted POS tags. Two thousand clusters produced highest

TABLE 5.4: Lexicalzed parsing results by applying word clustering.

Urdu text + Gold POS						
# Clusters	Measure	Small	Medium	Long	Accum.	
1K	*LR*	93.8	87.9	85.4	87.5	
	LP	91.2	85.9	84.4	85.9	
	F_1	92.5	86.9	84.9	**86.7**	
2K	*LR*	93.9	88.4	85.04	87.6	
	LP	91.6	86.4	84.18	86.1	
	F_1	92.8	87.4	84.61	**86.8**	
3K	*LR*	93.0	88.2	85.3	87.5	
	LP	90.8	86.3	84.5	86.0	
	F_1	91.9	87.2	84.9	**86.8**	
4K	*LR*	93.0	88.1	84.6	87.2	
	LP	90.3	85.8	83.3	85.3	
	F_1	91.6	86.9	83.9	**86.2**	
5K	*LR*	92.6	88.2	85.6	87.6	
	LP	90.7	86.3	84.8	86.2	
	F_1	91.6	87.2	85.2	**86.9**	
6K	*LR*	93.0	87.8	85.5	87.4	
	LP	90.8	86.0	84.8	86.0	
	F_1	91.9	86.9	85.1	**86.7**	
7K	*LR*	92.7	88.0	85.7	87.6	
	LP	90.4	86.1	84.8	86.0	
	F_1	91.5	87.1	85.2	**86.7**	
Urdu text + Predicted POS						
# Clusters	Measure	Small	Medium	Long	Accum.	POS
1K	*LR*	90.1	85.6	82.7	84.9	95.0%
	LP	88.0	83.6	81.3	83.1	
	F_1	89.0	84.6	81.98	**84.0**	
2K	*LR*	89.6	86.3	83.1	85.3	95.5%
	LP	87.8	84.2	81.9	83.6	
	F_1	88.7	85.2	82.5	**84.5**	
3K	*LR*	88.4	86.2	82.7	85.0	95.7%
	LP	86.2	84.2	81.5	83.3	
	F_1	87.3	85.2	82.1	**84.2**	
4K	*LR*	88.7	86.1	83.2	85.2	95.9%
	LP	86.3	83.8	81.7	83.2	
	F_1	87.5	84.9	82.4	**84.2**	
5K	*LR*	88.4	86.2	83.5	85.3	96.0%
	LP	85.6	84.0	82.4	83.5	
	F_1	87.0	85.1	82.9	**84.4**	
6K	*LR*	88.9	86.0	83.2	85.2	96.0%
	LP	86.3	83.7	81.9	83.3	
	F_1	87.6	84.9	82.5	**84.2**	
7K	*LR*	87.7	85.9	83.1	85.0	95.9%
	LP	85.2	83.7	81.8	83.1	
	F_1	86.4	84.8	82.5	**84.0**	

f-score of 84.5 with a POS accuracy of 95.5%. However, five thousands cluster provide an f-score of 84.4 with a POS tagging accuracy of 96% which is quite promising. Training with higher number of clusters resulted in lowing the scores. Therefore, it is concluded that five thousands clusters provide optimal parsing scores for a lexicalized parser on the CLE-UTB.

5.1.2 Data-oriented parsing

Table 5.5 presents the parsing results by training a data-oriented parser which is based on tree substitution grammars. The parser has been trained by using updated POS tags set and markovization values. We have further performed experiments with different tree heights starting from height one up to three.

The DOP parser produced best f-score of 87.1 when trained with maximum subtree of height one against gold POS tags. Training with heights two and three, the results decreased to 86.9. However, the predicted POS tags produce best f-scores when trained with maximum height of three. The POS accuracy for different heights remains almost similar. The DOP parser performs better than lexicalized parser as it caters the context by computing probability of subtrees including POS tags and lexical values.

The DOP parser was further evaluated against the ordered test set (Section 4.3.6) to analyze its ability to learn different word orders of the language and the coverage of the probable orders in the teebanks. Table 5.6 presents the DOP results on the ordered test set when trained with maximum subtree height of one.

Table 5.6 presents the parsing scores against three word orders SOV, OSV and SV. The results have been computed for gold and predicted POS tags. F-scores for gold

TABLE 5.5: Data-oriented parsing results against gold and predicted POS tags.

Parsing models	Urdu text + Gold POS				
	Measure	Small	Medium	Long	Accum.
DOP_maxheight=1	*LR*	93.1	88.7	84.4	87.4
+Markovization	*LP*	91.3	87.5	84.8	86.8
	F_1	92.2	88.1	84.6	**87.1**
DOP_maxheight=2	*LR*	94.0	88.0	84.5	87.2
+Markovization	*LP*	92.3	86.9	84.7	86.6
	F_1	93.1	87.4	84.6	**86.9**
DOP_maxheight=3	*LR*	93.8	88.3	84.1	87.2
+Markovization	*LP*	92.0	87.0	84.8	86.6
	F_1	92.9	87.7	84.5	**86.9**
	Urdu text + Predicted POS				
	Measure	Small	Medium	Long	Acc.	POS
DOP_maxheight=1	*LR*	87.9	85.0	80.0	83.3	95.5%
+Markovization	*LP*	87.0	84.3	81.7	83.6	
	F_1	87.5	84.7	80.8	**83.4**	
DOP_maxheight=2	*LR*	89.2	84.7	80.3	83.4	95.6%
+Markovization	*LP*	88.4	84.2	81.8	83.7	
	F_1	88.8	84.4	81.1	**83.5**	
DOP_maxheight=3	*LR*	88.2	84.8	80.4	83.4	95.5%
+Markovization	*LP*	87.7	84.5	82.1	83.9	
	F_1	88.0	84.6	81.2	**83.6**	

POS are quite high for all three word orders. SOV, OSV and SV orders provide f-scores of 92.9, 91.7 and 92.5 respectively. On the order hand, the DOP parser performs with f-scores of 87.8, 87.7 and 86.2 for SOV, OSV and SV orders receptively. The POS tagging accuracies are also promising. The results for ordered test set depict that the corpus has the coverage of probable word orders.

TABLE 5.6: Parsing results with SOV, OSV and SV word orders.

Parser	Urdu text + Gold POS				
	Order	*LR*	*LP*	F_1	
DOP_maxheight=1	SOV	94.0	91.7	92.9	
+Markovization	OSV	92.5	91.0	91.7	
	SV	93.5	91.5	92.5	
		Urdu text + Predicted			
	Order	*LR*	*LP*	F_1	POS
DOP_maxheight=1	SOV	88.4	87.2	87.8	96.0
+Markovization	OSV	88.2	87.1	87.7	95.7
	SV	87.3	85.2	86.2	94.5

5.1.3 Neural Parsing

Neural parsers use the powers of neural networks to provide the improved constituency parsing. Recursive neural network based parser has been trained on top 200 parse trees produced by a lexicalized PCFG parser and BiLSTM parser converts the parse trees into labels and performs sequence labeling by employing two LSTM layers. We have performed transfer learning along with both parsers which was helpful to improve the parsing scores. We have also included character embeddings with BiLSTM parser to capture the morphological information of the language. Character embeddings are contextual vectors achieved by the context of characters in the train set by using an LSTM layer. We further developed a BiLSTM network based POS tagger which performed with an accuracy of 96.3%. We also performed parsing by including functional tags to achieve the accuracy for grammatical relations along with phrase labels. Table 5.7 presents the results of neural parsers with and without transfer learning.

The RNN parser produced the accumulative f-scores of 87.1 and 86.4 with and

TABLE 5.7: Neural parsing results for gold and predicted POS tags.

Parsing models	Urdu text + Gold POS				
	Measure	Small	Medium	Long	Accum.
RNN Parser	*LR*	92.7	87.9	84.9	87.2
+No_emb.	*LP*	91.2	85.6	84.2	85.7
	F_1	91.9	86.7	84.5	**86.4**
RNN Parser	*LR*	92.2	88.2	86.0	87.7
+35M_WV	*LP*	90.7	86.4	85.6	86.5
	F_1	91.4	87.3	85.8	**87.1**
BiLSTM Parser	*LR*	87.4	81.9	78.0	80.9
+No_emb	*LP*	90.1	84.7	82.8	84.5
+Relative labels	F_1	88.7	83.3	80.3	**82.7**
BiLSTM Parser	*LR*	92.6	87.3	85.5	87.1
+35M_WV	*LP*	94.4	88.5	88.3	89.2
+Relative labels	F_1	93.5	87.9	86.9	**88.1**
BiLSTM Parser	*LR*	92.8	89.7	86.5	88.7
+35M_WV	*LP*	93.5	89.7	88.3	89.5
+Proposed labels	F_1	93.2	89.7	87.4	**89.1**
	Urdu text + Predicted POS				
	Measure	Small	Medium	Long	Accum.	POS
RNN Parser	*LR*	88.1	85.1	81.2	83.9	95.0
+No_emb.	*LP*	87.1	84.0	81.4	83.3	
	F_1	87.6	84.6	81.3	**83.6**	
RNN Parser	*LR*	86.4	85.8	82.0	84.3	95.3
+35M_WV	*LP*	86.6	84.9	82.4	84.0	
	F_1	86.5	85.3	82.2	**84.2**	
BiLSTM Parser	*LR*	84.4	80.9	77.8	80.0	96.3
+No_emb	*LP*	86.0	82.1	79.9	81.6	
+Relative labels	F_1	85.2	81.5	78.8	**80.8**	
BiLSTM Parser	*LR*	89.9	86.6	83.5	85.7	96.3
+35M_WV	*LP*	90.7	87.2	85.1	86.7	
+Relative labels	F_1	90.3	86.9	84.3	**86.2**	
BiLSTM Parser	*LR*	91.6	87.5	84.8	86.8	96.3
+35M_WV	*LP*	92.9	88.3	86.9	88.2	
+Proposed labels	F_1	92.3	87.9	85.8	**87.5**	

without using pre-trained word embeddings. Our BiLSTM parser outperforms the RNN parser by using word embeddings with an f-score of 88.1. However, it produces low scores when trained without pre-trained word embeddings and gives an accumulative

f-score of 82.7. The transfer learning improved the f-score by 5.4 points. For RNN parser the difference is of 0.7 points. By using the predicted POS tags, the BiLSTM parser performed with a best f-score of 86.2 which is highest amongst our previously trained models. The RNN parser uses pre-cached parse trees achieved from a lexicalized grammar based parser and BiLSTM parser performs the sequence labeling by using the sentences from the treebank. However, the performance of RNN parser is equal to the DOP parser.

We further proposed a sequential labeling for the Urdu treebank and trained the BiLSTM parser along with the pre-trained word embeddings. Our proposed labeling offered significantly lower number of labels hence helped to improve parsing scores. Similar to other models, it has been evaluated against small, medium and long sentences. The proposed labeling outperforms relative labels and it produced a best f-score of 89.1 which is one percent higher than 88.1 the previous best f-score. On the other hand, the model has been evaluated by using predicted POS tags and it achieved an f-score of 87.5 as compared to the previous best of 86.2. The f-score improvements are also observed for all three groups of the test set.

We further performed the analysis of parsing results by achieving the scores with respect to phrase labels. Table 5.8 shows phrase-wise parsing results from the BiLSTM parser by using the proposed labeling against gold and predicted POS tags.

The highest parsing scores are achieved for VC phrase which are 97.8 and 95.5 for gold and predicted POS tags respectively. The second highest scores are achieved for PP phrases which are 89.6 and 87.6. Similarly, NP phrase produced f-scores of 87.2 and 84.2. The phrase PREP and DMP produce lower f-scores. The reason is the low

TABLE 5.8: Parsing results of the BiLSTM parser with respect to phrase labels.

	Urdu text + Gold POS			Urdu text + Predicted POS		
Phrase labels	*LR*	*LP*	F_1	*LR*	*LP*	F_1
NP	87.4	86.9	87.2	84.6	83.9	84.2
PP	88.1	91.1	89.6	86.1	89.1	87.6
S	84.2	86.8	85.5	83.5	87.2	85.3
VC	98.3	97.3	97.8	95.7	95.3	95.5
SBAR	83.0	82.4	82.7	81.5	83.5	82.5
ADJP	72.6	77.2	74.8	64.4	77.5	70.4
ADVP	87.8	86.7	87.2	81.0	79.3	80.2
QP	69.4	70.8	70.1	65.3	72.7	68.8
FFP	71.4	78.9	75.0	61.9	65.0	63.4
DMP	20.0	100.0	33.3	20.0	12.5	15.4
PREP	0	nan	nan	0	nan	nan

frequency of these labels in the train and test sets. Table 5.9 further shows the confusion values amongst different phrase labels. However, the patterns are quite similar for both cases. The counts are ordered in decreasing order. By looking at the confusion values when evaluated with gold POS tags, the highest confusion was among NP and PP labels. In this case, the gold label was PP and predicted label was NP and this occurred 46 times. Similarly, PP was confused with NP for just two times. The syntactic structure of PP phrases is more predictable as compared to NPs. A PP phrase contains an NP followed by a post-position with a POS tag of PSP. There was a confusion between NP and S labels 16 times however, S label was predicted against NP for 11 times. The ADJP label was confused by the parser with NP 14 times whereas the NP was confused with ADJP for only five times. The reason is syntactic structure of the ADJP phrase as they contain adjectives with POS tag of JJ in them. The label S was confused with PP nine times and PP was not confused with S in the results when included gold POS tags in the training.

In the same fashion, Table 5.9 shows the values against all the labels when

TABLE 5.9: Category Statistics (all categories / errors) with respect to reference corpus.

Urdu text + Gold POS			Urdu text + Predicted POS		
Candidate	Gold	Count	Candidate	Gold	Count
NP	PP	46	NP	PP	48
NP	S	16	NP	ADJP	35
ADJP	NP	14	NP	S	22
S	NP	11	ADJP	NP	18
S	PP	9	NP	ADVP	13
NP	ADJP	5	S	PP	10
VC	ADVP	4	ADVP	NP	9
QP	NP	4	VC	NP	8
NP	ADVP	4	S	NP	8
SBAR	S	4	QP	NP	7
NP	QP	3	NP	QP	5
S	ADJP	3	VC	ADVP	4
VC	NP	2	S	SBAR	4
NP	PREP	2	S	ADJP	3
VC	S	2	PP	S	3
PP	NP	2	NP	FFP	3
QP	ADVP	1	PP	NP	3
ADVP	S	1	SBAR	S	3
DMP	ADJP	1	ADVP	VC	3
ADVP	PP	1	NP	PREP	2
PP	ADJP	1	VC	S	2
NP	FFP	1	ADVP	ADJP	2
ADVP	QP	1	NP	VC	2
QP	S	1	ADVP	QP	2
QP	ADJP	1	QP	ADVP	1
S	SBAR	1	SBAR	ADJP	1
ADJP	PP	1	VC	ADJP	1
SBAR	PP	1	S	VC	1
ADVP	NP	1	NP	SBAR	1
–	–	–	ADVP	PP	1
–	–	–	PP	ADJP	1
–	–	–	ADJP	VC	1
–	–	–	S	ADVP	1
–	–	–	QP	S	1
–	–	–	DMP	ADVP	1
–	–	–	FFP	ADJP	1
–	–	–	FFP	NP	1
–	–	–	SBAR	PP	1
–	–	–	ADVP	S	1

achieved results from our parser. The evaluation by using predicted tags also produce similar values against phrase labels.

We have further experimented with BiLSTM parser to parse the functional labels. The functional labels provide the information of grammatical roles in a sentences. As we have seen that our corpus has a coverage of frequent word orders, it is important to learn functional labels. Table 5.10 shows the parsing results along with functional accuracies when trained on the best performing BiLSTM parser. We have trained the parser by using relative labels as well as the proposed labeling.

TABLE 5.10: Neural parsing results by including functional labels.

Parsing models	Urdu text + Gold POS					
	Measure	Small	Medium	Long	Accum.	
BiLSTM Parser	*LR*	91.5	86.5	83.1	85.7	
+35M_WV	*LP*	93.2	88.5	85.6	87.8	
+Relative labels	F_1	92.4	87.5	84.4	**86.7**	
	Func.	81.3%	82.2%	81.7%	81.9%	
BiLSTM Parser	*LR*	92.8	88.5	84.8	87.5	
+35M_WV	*LP*	93.4	89.4	87.1	88.9	
+Proposed labels	F_1	93.1	89.0	85.9	**88.2**	
	Func.	80.5%	83.3%	82.5%	**82.7%**	
	Urdu text + Predicted POS					
	Measure	Small	Medium	Long	Accum.	POS
BiLSTM Parser	*LR*	89.3	85.5	83.5	85.1	96.3%
+35M_WV	*LP*	91.8	87.1	85.5	87.0	
+Relative labels	F_1	90.5	86.3	84.5	**86.0**	
	Func.	82.3%	84.0%	84.2%	83.8%	
BiLSTM Parser	*LR*	91.5	87.0	82.5	85.6	
+35M_WV	*LP*	91.9	87.5	85.1	87.0	
+Proposed labels	F_1	91.7	87.2	83.8	**86.3**	
	Func.	80.7%	83.4%	82.1%	**82.6%**	

The parsing results with relative labels as f-scores of constituency parsing are

86.7 and 86.0 against gold and predicted tags. Similarly, the accuracies for functional labels are 81.9% and 83.8% which are quite promising when achieved from a constituency parser trained on a language with flexible word order. However, the proposed labels, which were trained on the same configurations of the parser, outperform the relative labels by producing state of the art functional results for the CLE-UTB. The parser produced f-score of 88.2 and 86.2 against gold and predicted POS tags. The functional accuracies are 82.7% and 82.6%, which are promising.

Table 5.11 presents accuracies of individual functional labels. The label 'G' which marks the genitive post-positional phrase produces the highest f-score of 99.1% against both gold and predicted POS tags. The SUBJ label produces f-scores of 85.1% and 84.5%. The POF label which has been used to mark complex predicate structures, performs with accuracies of 83.7% and 82.8%. The labels INJ and OBL produce low accuracies. The INJ has less occurrences in the corpus while the OBL has been used to mark PP phrases as well as NPs. Table 5.12 shows the error analysis by presenting the predicted confusions among the functional labels.

TABLE 5.11: Parsing results of BiLSTM parser with respect to individual functional labels

	Urdu text + Gold POS			Urdu text + Predicted POS		
Functional labels	*LR*	*LP*	F_1	*LR*	*LP*	F_1
SUBJ	85.9	84.1	85.1	82.2	86.8	84.5
G	99.5	98.7	99.1	99.7	98.5	99.1
POF	84.6	82.9	83.7	83.7	82.0	82.8
OBJ	61.2	61.5	61.4	63.2	60.1	61.6
ADJ	82.5	83.0	82.8	84.6	80.5	82.5
PDL	76.4	82.1	79.2	81.1	81.1	81.1
OBL	28.7	40.8	33.7	32.5	44.7	37.6
VOC	66.7	88.9	76.2	56.5	76.5	65.0
VALA	100	87.5	93.3	90.0	90.0	90.0
INJ	80.0	66.7	72.7	100	100	100

The confusion results are also computed against functional labels with respect. It is important to note the functional accuracies are quite similar for both gold and predicted tags. First two rows of Table 5.12 show the confusions among SUBJ and OBJ labels. These both labels are used to mark grammatical roles with PPs and NPs. In case of PPs, the syntactic structure is quite predictable. For example, a subject PP contains an NP followed by an ergative case maker *nE* which has a POS tag of PSP-E. In case of a PP object, it contains an NP followed by an accusative case marker *kO* with a tag PSP-A. On the other hand, subject and object NPs are annotated in a similar manner resulting a fewer confusions. The labels POF, PDL, OBL and ADJ are also marked with NPs and caused some confusions in the learning process. However, POF and PDL are also marked with ADJPs and QPs. The table further shows all the confused labels and their counts while trained with the BiLSTM parser. Overall, the accuracy of functional labels is quite promising.

5.1.4 Summary of Results

Detailed results have been presented in this chapter which show precision, recall and f-scores for all the trained parsing models. This section summarizes the results by presenting these measures for accumulative scores. Table 5.13 shows the best parsing results from all models.

Table 5.13 presents the parsing results of best performing models against the whole test set. The F_1 scores are appearing as bold faced numbers. Overall, the results show upwards trends. We started the experiments from base-line probabilistic context-free grammars (PCFGs) towards the state of the art neural parsing for the CLE-UTB.

TABLE 5.12: Error analysis of functional labels with respect to reference corpus.

Urdu text + Gold POS			Urdu text + Predicted POS		
Candidate	Gold	Count	Candidate	Gold	Count
SUBJ	OBJ	66	OBJ	SUBJ	74
OBJ	SUBJ	56	SUBJ	OBJ	51
OBJ	POF	30	POF	OBJ	39
POF	OBJ	28	OBJ	POF	30
POF	PDL	27	POF	PDL	24
SUBJ	PDL	23	PDL	SUBJ	21
PDL	POF	17	SUBJ	PDL	16
OBJ	OBL	17	PDL	POF	14
SUBJ	POF	16	OBJ	OBL	14
OBL	OBJ	16	POF	SUBJ	10
PDL	SUBJ	14	SUBJ	POF	10
OBL	SUBJ	11	OBL	OBJ	8
POF	SUBJ	10	ADJ	POF	8
OBL	ADJ	9	OBL	SUBJ	8
ADJ	OBL	8	ADJ	SUBJ	7
SUBJ	OBL	7	ADJ	OBL	7
SUBJ	ADJ	6	POF	ADJ	6
ADJ	OBJ	5	PDL	OBL	5
ADJ	SUBJ	5	OBL	ADJ	5
PDL	OBJ	4	SUBJ	ADJ	5
ADJ	POF	4	SUBJ	OBL	5
POF	ADJ	4	OBJ	ADJ	4
PDL	OBL	3	POF	OBL	4
OBJ	PDL	3	PDL	OBJ	3
ADJ	PDL	3	ADJ	OBJ	2
OBJ	ADJ	2	OBJ	PDL	2
POF	OBL	2	G	SUBJ	1
G	SUBJ	1	PDL	ADJ	1
PDL	ADJ	1	OBL	PDL	1
SUBJ	VOC	1	VOC	SUBJ	1
OBL	POF	1	PDL	G	1
–	–	–	ADJ	PDL	1
–	–	–	OBL	POF	1
–	–	–	SUBJ	VOC	1

The PCFG based model performed with base-line scores of 76.2 and 71.4 against gold POS and predicted POS tags respectively. By incorporating an updated POS tagset as

TABLE 5.13: Summary of parsing results with respect accumulative scores of best performing models.

Parsing models	Gold POS			Predicted POS			
	LR	*LP*	F_1	*LR*	*LP*	F_1	POS Acc.
Baseline PCFG	75.8	76.7	**76.2**	70.6	72.2	**71.4**	94.5%
PCFG + POS2	82.7	83.6	**83.1**	78.2	79.7	**78.9**	94.9%
PCFG + POS2 + Markov.	84.9	84.9	**84.9**	80.1	81.0	**80.5**	95.1%
Lex-PCFG + Urdu heads	87.1	85.1	**86.1**	84.2	82.2	**83.2**	95.7%
Lex-PCFG + Lemmas	87.2	85.2	**86.2**	84.4	82.4	**83.4**	95.1%
Lex-PCFG + Clusters(5K)	87.6	86.2	**86.9**	85.3	83.5	**84.4**	96.0%
DOP + h=1 + Markov.	87.4	86.8	**87.1**	83.3	83.6	**83.4**	95.5%
RNN Parser + Emb.	87.7	86.5	**87.1**	84.3	84.0	**84.2**	95.3%
BiLSTM Parser + Emb.	87.1	89.2	**88.1**	85.7	86.7	**86.2**	96.3%
BiLSTM Parser + Emb. + Proposed labels	88.7	89.5	**89.1**	86.8	88.2	**87.5**	**96.3**%

POS2, these scores were significantly increased to 83.1 and 78.9. The markovization was also helpful to increase the score by including contextual information in the context-free grammars and the improved scores are 84.9 and 80.5.

The lexicalized grammar based models have been experimented along with different linguistic features including an Urdu head model, lemmatization and word clustering. Different number of word clusters have been used for training starting from one thousand to seven thousands clusters and the model performs well on five thousands word cluster labels and performed with f-scores of 86.9 and 84.4. The data-oriented parsing (DOP) and recursive neural network (RNN) based parsers also perform well on the CLE-UTB. The RNN based model produced f-scores of 87.1 and 84.2.

However, the proposed parser which is based on bidirectional long-short term

memory (BiLSTM) networks, performed with the state of the art scores on the CLE-UTB. We also developed a POS tagger based on BiLSTM networks and the tagging has been done for our neural model. We trained the parsing model by using an existing relative labeling and achieved f-scores of 88.1 and 86.2. We further proposed a labeling technique for the CLE-UTB which is based on the syntactic structure and annotation of the treebank. The proposed labels improved the parsing scores significantly and produced f-scores of 89.1 and 87.5 which are quite promising for a morphologically-rich and free word order language.

5.1.5 Discussions

Tsarfaty et al. [148] raised three questions and their answers are crucial to parse morphologically rich languages. The first question is about the language representation and input type of the data set. We have used three types of inputs and language representations for training and testing the statistical parsers. The first input representation was the text written in Urdu script along with POS tags. All the parsers have been evaluated against gold and predicted POS tags. The grammar-based parser which uses the lexical information performed with f-scores of 86.1 and 83.2 including gold and predicted POS tags respectively. The same representation has been trained with DOP and neural parsers. DOP parser performed parsing with best f-scores of 87.1 and 83.6 and our BiLSTM parser along with relative labeling produced the parsing results with f-scores of 88.1 and 86.2 by using gold and predicted POS respectively. We further derived an updated sequence labeling for the CLE-UTB which produced state of the art parsing results with f-scores of 89.1 and 87.5.

The morphologically rich nature of Urdu could cause the issue of data sparsity. To cater this property of the language, we have used two other representations. The second representation of the corpus contained lemmas for each word. For that purpose, we implemented a lemmatizer which mapped each word on its root form in the corpus. The lemmatized dataset produced a subtle improvement in the parsing results. The parsing accuracy is dependent on the accuracy of the lemmatizer. The third representation produced word clusters and replaced the words with their cluster tags. The clusters tags were achieved with respect to syntactic similarities of words in the corpus. We experimented with clustered data set by training the same lexicalized parser. The best parsing results were produced by five thousand clusters with f-scores of 86.9 and 84.4 for gold and predicted POS tags. The clustered dataset produced the improvements of 0.8 and 0.6 points as compared to plain Urdu text. However, the neural parser based on bi-directional LSTM networks produced top of the list results and catered the morphological aspect by implementing character embeddings during training of the networks. The neural parsers further used the transfer learning by including pre-trained word representations which were also helpful to learn the surface forms and out of vocabulary words. Our parsing experiments and results are evident that the input type and language representations are important to improve the parsing scores for morphologically-rich languages.

The second question raised by Tsarfaty et al. [148] is about influence of morphology on POS and phrase labels. A POS tags should have the ability to mark all word classes of a language with sufficient number of tags. A morphologically-rich language may have a large number of surface forms which makes it impractical to have separate

tag for each surface form. The number of tags should be sufficiently enough to mark the words of such languages because huge number of tags will require more data to train statistical taggers. The POS tagger which has been used for the annotation of the CLE-UTB contains 35 tags. It marks the main classes of open class words by one tag. For example, it marks nouns, adjectives, adverbs and verbs by using single tag for each. The tag set reduced the data requirement while keeping its effectiveness for training a statistical tagger.

Similarly, the phrase annotation is relatively flat and the label set contains 11 phrase labels. This label set has been derived from a universal label set which proposed abstract labels by merging related labels. However, the label set contains sufficient number of tags to mark syntactic structure of the language. To perform the statistical parsing for the CLE-UTB, we further derived additional representations of tags set as well. We derived an updated POS tags set (POS2) by syntactic categorization of post-positions and punctuations. The updated tags helped to improve the parsing results by a factor of seven points for the baseline PCFG parser. All later experiments were performed by using updated POS tag set. Phrase label set was further extended by using vertical and horizontal markovizations. The markovizations help to include contextual information in the parse trees. This representation improved the parsing results by 1.8 points and they have been used with all grammar-based parsers. Our annotation scheme also has ten functional labels to mark grammatical roles. Parsing with functional labels was also helpful to include contextual and grammatical information in the parse trees and the BiLSTM parser produced quite satisfactory parsing scores and functional accuracies. Simplified annotation label sets are helpful to annotate and parse morphologically-rich

languages.

The third question raised by Tsarfaty et al. [148] is about the size of dataset. The size of the dataset depends on the label sets and the coverage of surface forms of a morphologically-rich language. The data requirements can be reduced by using a linguistically well-defined label set. We have used compact POS and phrase label sets for the annotation of the CLE-UTB which were further updated for statistical parsing. The treebank size is quiet sufficient to train statistical parsers. It has the coverage of probable word orders. The parsing results on ordered test set are quite promising to claim that the treebank has the coverage of probable word orders. Similarly, our BiLSTM POS tagger performed with an accuracy of 96.3% which is quite acceptable. Transfer learning helped to provide the morphological coverage in the statistical training to overcome the issues of data sparsity and out of vocabulary words and produced state of the art parsing results on the CLE-UTB.

This research presented the development of a phrase structure treebank and comprehensive comparison of parsing methods. The parsing techniques include grammar-based and neural parsers. Grammar-based parsers include simple PCFG parser, lexicalized PCFG parser and tree-substitution grammars and neural parsers contain an RNN parser and a BiLSTM parser. Additionally, several linguistic features have been derived and trained to achieve improved results. These features include updated POS tags, lemmatization, word clustering and markovizations. Lexicalized PCFG parser performs with best f-scores of 86.9 and 84.4 when evaluated by using gold and predicted POS tags. These scores were achieved by training word clusters with five thousand cluster labels. Data-oriented parsing, based on tree-substitution grammars, outperformed the

lexicalized parser when evaluated on gold POS tags and produces best f-score of 87.1. The RNN parser gives competitive parsing score with comparison to data-oriented parsing. However, our BiLSTM parser outperforms the grammar-based and RNN parsers and produced f-scores of 88.1 and 86.1 when evaluated with gold and predicted POS tags respectively. Pre-trained word representations help to provide significant improvements in the results by transfer learning. The BiLSTM parser also performs better when trained with functional labels and provides a best functional accuracy of 83.8% while keeping the f-score around 86. The three groups of the test set are helpful to analyze the parsing results. All the parsers perform well on smaller sentences as compared to longer ones. Overall, the BiLSTM parser and POS tagger outperform grammar-based parsers.

6. CONCLUSIONS

This research presents the development of a phrase structure treebank (CLE-UTB) for Urdu. The annotation scheme has been derived from various sources to make it compatible with existing treebanks. The annotation is suitable for a consistent phrase structure of Urdu being a free word order language. A balanced corpus containing text from a number of text domains, has been used. The functional labels were used to mark the grammatical relations in the treebank which added a layer on the phrase labels. The phrase annotation shows the capability for the annotation of several linguistic aspects of Urdu like flexible argument structure, complex predication and case system.

Several processes have been applied to perform treebank evaluation during the annotation. At the first step, completeness and correctness were checked which was followed by a manual revision of the whole corpus. The second step was performed semi-automatically to identify linguistically implausible structures. For that purpose, we devised a grammar-based evaluation to identify implausible grammar rules. The method reported the constituents with low frequency in the treebank. The reported constituents were further reviewed and corrected. At the third step, an automatic consistency checking tool was used to check context-based phrasal inconsistencies. The checker identified outliers in the form of a report which were further corrected after review. The inter-annotator agreement was above 90 which was calculated by using a

reference corpus.

For statistical parsing, several parsers have been training including grammar-based parsers, data-oriented and neural network based models. Additionally, syntactic features have been used like POS sub-categorization, lemmatization, markovization and word clusters. The language representation, input types and phrase labeling are crucial to parse a morphologically-rich language. Lemmatization and word cluster were helpful to improve the parsing scores of lexicalized parsing. However, neural parsers outperformed the grammar-based parsers by using transfer learning. The word embeddings were used for neural parsing which were trained on a large plain text corpus. External word embeddings provided a large range of vocabulary as well as their semantic information. Word embeddings reduced the effect of data sparsity cause by the morphologically-rich nature of the language. A bidirectional long-short term memory based parser was trained on the final version of the treebank. Our parser produced an f-score of 89.1 which is quite satisfactory for a morphologically rich and a free word order language. Transfer leaning helps to achieve better statistical parsing for a morphologically-rich languages.

The phrase annotation of our treebank is compatible with dependency structure due to flat annotation scheme and functional labeling. We have automatically converted the phrase structure into dependency structure. Phrasal head rules and dependency label mappings were devised to find correct dependencies. Furthermore, the dependency parsing has been performed by using MaltParser and a neural dependency parser. However, dependency conversion and parsing are additional tasks which will be matured in future.

6.1 FUTURE WORK

The dependency conversion is still in progress. In future, we will perform a comprehensive evaluation of the treebank by using an annotated reference corpus. We will study the annotation and structural compatibility of the dependency treebank with existing resources.

BIBLIOGRAPHY

[1] Qaiser Abbas. Building a Hierarchical Annotated Corpus of Urdu: The URDU.KON-TB Treebank. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 66–79. Springer, 2012.

[2] Qaiser Abbas. Building computational resources: The URDU.KON-TB treebank and the Urdu parser. *Konstanzer Online-Publication-System (KOPS)*, 2014.

[3] Anne Abeillé and Nicolas Barrier. Enriching a French Treebank. In *LREC*, 2004.

[4] Anne Abeillé, Lionel Clément, and François Toussenel. Building a treebank for French. In *Treebanks*, pages 165–187. Springer, 2003.

[5] Steven Abney, S Flickenger, Claudia Gdaniec, C Grishman, Philip Harrison, Donald Hindle, Robert Ingria, Frederick Jelinek, Judith Klavans, Mark Liberman, et al. Procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the workshop on Speech and Natural Language*, pages 306–311. Association for Computational Linguistics, 1991.

[6] Susana Afonso, Eckhard Bick, Renato Haber, and Diana Santos. Floresta sintá (c) tica: a Treebank for Portuguese. In *quot; In Manuel González Rodrigues; Carmen Paz Suarez Araujo (ed) Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002)(Las Palmas de Gran Canaria Espanha 29-31 de Maio de 2002) Paris: ELRA*. ELRA, 2002.

[7] Tafseer Ahmed and Miriam Butt. Discovering Semantic Classes for Urdu N-V Complex Predicates. In *Proceedings of the Ninth International Conference on Computational Semantics (IWCS 2011)*, 2011.

[8] Misbah Akram and Sarmad Hussain. Word segmentation for Urdu OCR system. In *Proceedings of the 8th Workshop on Asian Language Resources, Beijing, China*, pages 88–94, 2010.

[9] Juan Aparicio, Mariona Taulé, and Maria Antònia Martí. AnCora-Verb: A Lexical Resource for the Semantic Annotation of Corpora. In *LREC*, 2008.

[10] Justin Bayer and Christian Osendorfer. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*, 2014.

[11] Akshar Bharati, Vineet Chaitanya, Rajeev Sangal, and KV Ramakrishnamacharyulu. *Natural Language Processing: A Paninian Perspective*. Prentice-Hall of India New Delhi, 1995.

[12] Akshar Bharati, Rajeev Sangal, and Dipti M Sharma. SSF: Shakti Standard Format Guide. *Language Technologies Research Centre, International Institute of Information Technology, Hyderabad, India*, pages 1–25, 2007.

[13] Riyaz Ahmad Bhat, Irshad Ahmad Bhat, and Dipti Misra Sharma. Improving transition-based dependency parsing of Hindi and Urdu by modeling syntactically relevant phenomena. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 16(3):17, 2017.

[14] Riyaz Ahmad Bhat, Rajesh Bhatt, Annahita Farudi, Prescott Klassen, Bhuvana Narasimhan, Martha Palmer, Owen Rambow, Dipti Misra Sharma, Ashwini Vaidya, Sri Ramagurumurthy Vishnu, et al. The Hindi/Urdu Treebank Project. pages 659–697, 2017.

[15] Rajesh Bhatt, Annahita Farudi, and Owen Rambow. *Hindi-Urdu Phrase Structure Annotation Guidelines*. 2013.

[16] Ann Bies, Mark Ferguson, Karen Katz, Robert MacIntyre, Victoria Tredinnick, Grace Kim, Mary Ann Marcinkiewicz, and Britta Schasberger. Bracketing Guidelines for Treebank II Style Penn Treebank Project. *University of Pennsylvania*, 97:100, 1995.

[17] Daniel M Bikel. Design of a Multi-Lingual, Parallel-Processing Statistical Parsing Engine. In *Proceedings of the Second International Conference on Human Language Technology Research*, pages 178–182. Citeseer, 2002.

[18] Haris Bin Zia, Agha Ali Raza, and Awais Athar. Urdu Word Segmentation using Conditional Random Fields (CRFs). In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2562–2569, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/C18-1217`.

[19] Ezra Black, Steven Abney, Dan Flickinger, Claudia Gdaniec, Ralph Grishman, Phil Harrison, Donald Hindle, Robert Ingria, Frederick Jelinek, Judith L Klavans, et al. A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars. In *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991*, 1991.

[20] Rens Bod. A computational model of language performance: Data oriented parsing. In *Proceedings of the 14th conference on Computational linguistics-Volume 3*, pages 855–859. Association for Computational Linguistics, 1992.

[21] Tina Bögel and Miriam Butt. Possessive clitics and ezafe in Urdu. *Morphosyntactic Categories and the Expression of Possession*, 199(291):86–129, 2013.

[22] Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. The TIGER Treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, volume 168, 2002.

[23] Miriam Butt. *The Structure of Complex Predicates in Urdu*. Center for the Study of Language (CSLI), 1995.

[24] Miriam Butt. *Theories of case*. Cambridge University Press, 2006.

[25] Miriam Butt and Tracy Holloway King. Urdu and the Parallel Grammar Project. In *Proceedings of the 3rd workshop on Asian language resources and international standardization-Volume 12*, pages 1–3. Association for Computational Linguistics, 2002.

[26] Miriam Butt and Tracy Holloway King. The Status of Case. In *Clause structure in South Asian languages*, pages 153–198. Springer, 2004.

[27] Miriam Butt and Tracy Holloway King. Questions and Information Structure in Urdu/Hindi. In *Proceedings of the LFG14 Conference*, pages 158–178. Stanford: CSLI Publications, 2014.

[28] Miriam Butt and Gillian Ramchand. Complex Aspectual Structure in Hindi/Urdu. *M. Liakata, B. Jensen, & D. Maillat, Eds The Syntax of Aspect*, pages 1–30, 2001.

[29] Miriam Butt, Helge Dyvik, Tracy Holloway King, Hiroshi Masuichi, and Christian Rohrer. The Parallel Grammar Project. In *COLING-02: Grammar Engineering and Evaluation*, 2002.

[30] Miriam Butt, Tina Bögel, Annette Hautli, Sebastian Sulger, and Tafseer Ahmed. Identifying Urdu complex Predication via Bigram Extraction. In *International Conference on Computational Linguistics*, pages 409–424, 2012.

[31] Eugene Charniak. Statistical Parsing with a Context-Free Grammar and Word Statistics. *AAAI/IAAI*, 2005(598-603):18, 1997.

[32] Danqi Chen and Christopher Manning. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

[33] Noam Chomsky. *The minimalist program*. MIT press, 2014.

[34] Montserrat Civit and Ma Antònia Martí. Building Cast3LB: A Spanish Treebank. *Research on Language and Computation*, 2(4):549–574, 2004.

[35] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[36] Michael Collins. Three Generative, Lexicalised Models for Statistical Parsing. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 16–23. Association for Computational Linguistics, 1997.

[37] Michael Collins. Head-Driven Models for Natural Language Parsing. *Ph.D. Thesis, Dept. of Computer and Information Science, University of Pennsylvania*, 1999.

[38] Michael Collins. Head-driven Statistical Models for Natural Language Parsing. *Computational linguistics*, 29(4):589–637, 2003.

[39] Michael Collins, Lance Ramshaw, Jan Hajič, and Christoph Tillmann. A Statistical Parser for Czech. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 505–512. Association for Computational Linguistics, 1999.

[40] Michael John Collins. A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 184–191. Association for Computational Linguistics, 1996.

[41] Anna Corazza, Alberto Lavelli, Giogio Satta, and Roberto Zanoli. Analyzing an Italian Treebank with State-of-the-Art Statistical Parsers. In *Proceedings of the Third Third Workshop on Treebanks and Linguistic Theories (TLT 2004)*, volume 1, page 155, 2004.

[42] Rodolfo Corona, Jesse Thomason, and Raymond Mooney. Improving black-box speech recognition using semantic parsing. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 122–127, 2017.

[43] Arnoldo Nunes da Silva, Osvaldo de Souza, and José Neuman de Souza. Sentiment parser based on X-Bar theory to Brazilian Portuguese. In *2020 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*, pages 166–171. IEEE, 2020.

[44] Marie-Catherine De Marneffe and Christopher D Manning. The Stanford Typed Dependencies Representation. In *Coling 2008: proceedings of the workshop on cross-framework and cross-domain parser evaluation*, pages 1–8, 2008.

[45] Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. Generating Typed Dependency Parses from Phrase Structure Parses. In *Lrec*, volume 6, pages 449–454, 2006.

[46] Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. Universal Stanford Dependencies: A Cross-Linguistic Typology. In *LREC*, volume 14, pages 4585–4592, 2014.

[47] Jon Dehdari, Lamia Tounsi, and Josef van Genabith. Morphological Features for Parsing Morphologically-Rich Languages: A Case of Arabic. In *Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich Languages*, pages 12–21, 2011.

[48] Jon Dehdari, Liling Tan, and Josef van Genabith. BIRA: Improved Predictive Exchange Word Clustering. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1169–1174, 2016.

[49] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for language Understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[50] Amit Dubey. What to Do When Lexicalization Fails: Parsing German with Suffix Analysis and Smoothing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 314–321. Association for Computational Linguistics, 2005.

[51] Amit Dubey and Frank Keller. Probabilistic Parsing for German Using Sister-Head Dependencies. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 96–103. Association for Computational Linguistics, 2003.

[52] Nadir Durrani and Sarmad Hussain. Urdu word segmentation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 528–536. Association for Computational Linguistics, 2010.

[53] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A Smith. Recurrent Neural Network Grammars. *arXiv preprint arXiv:1602.07776*, 2016.

[54] Toqeer Ehsan and Miriam Butt. Dependency Parsing for Urdu: Resources, Conversions and Learning. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 5202–5207, 2020.

[55] Toqeer Ehsan and Sarmad Hussain. Development and Evaluation of an Urdu Treebank (CLE-UTB) and a Statistical Parser. *Language Resources and Evaluation*, Jul 2020. ISSN 1574-0218. doi: 10.1007/s10579-020-09492-7. URL `https://doi.org/10.1007/s10579-020-09492-7`.

[56] Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. Learning to parse and translate improves neural machine translation. *arXiv preprint arXiv:1702.03525*, 2017.

[57] Ryan Gabbard, Seth Kulick, and Mitch Marcus. Fully Parsing the Penn Treebank. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 184–191, 2006.

[58] Charlotte Galves and Pablo Faria. Tycho Brahe Parsed Corpus of Historical Portuguese. *URL: http://www. tycho. iel. unicamp. br/tycho/corpus/en/index. html*, 2010.

[59] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Continual prediction using LSTM with forget gates. In *Neural Nets WIRN Vietri-99*, pages 133–138. Springer, 1999.

[60] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. In *9th International Conference on Artificial Neural Networks: ICANN '99*, pages 850–855. IET, 1999.

[61] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*, 3(Aug):115–143, 2002.

[62] Carlos Gómez-Rodríguez and David Vilares. Constituent Parsing as Sequence Labeling. In *Conference on Empirical Methods in Natural Language Processing, EMNLP2018*, pages 1314—-1324. Association for Computational Linguistics, 2018.

[63] Anne Göhring. Spanish Expansion of a Parallel Treebank. *Lizentiatsarbeit, University of Zurich*, 2009.

[64] Yoav Goldberg and Joakim Nivre. A dynamic oracle for arc-eager dependency parsing. *Proceedings of COLING 2012*, pages 959–976, 2012.

[65] Joshua Goodman. Probabilistic Feature Grammars. In *Advances in Probabilistic and Other Parsing Technologies*, pages 63–84. Springer, 2000.

[66] Joshua Goodman, Rens Bod, and Remko Scha. Efficient parsing of DOP with PCFG-reductions. 2003.

[67] JT Goodman. A BIT of Progress in Language Modeling Extended Version. *Machine Learning and Applied Statistics Group Microsoft Research. Technical Report, MSR-TR-2001-72*, 2001.

[68] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional LSTM networks for improved phoneme classification and recognition. In *International Conference on Artificial Neural Networks*, pages 799–804. Springer, 2005.

[69] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[70] Haohan Guo, Frank K Soong, Lei He, and Lei Xie. Exploiting syntactic features in a parsed tree to improve end-to-end TTS. *arXiv preprint arXiv:1904.04764*, 2019.

[71] Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. Semantic parsing for task oriented dialog using hierarchical representations. *arXiv preprint arXiv:1810.07942*, 2018.

[72] Jan Hajič, Eva Hajičová, Marie Mikulová, and Jiří Mírovskỳ. Prague dependency treebank. In *Handbook of Linguistic Annotation*, pages 555–594. Springer, 2017.

[73] Aaron Li-Feng Han, Derek F Wong, Lidia S Chao, Yi Lu, Liangye He, and Liang Tian. A Universal Phrase Tagset for Multilingual Treebanks. pages 247–258, 2014.

[74] Chung-hye Han, Na-Rae Han, and Eon-Suk Ko. Bracketing Guidelines for Penn Korean Treebank. *IRCS Technical Reports Series*, page 26, 2001.

[75] Chung-hye Han, Na-Rae Han, Eon-Suk Ko, Martha Palmer, and Heejong Yi. Penn Korean Treebank: Development and Evaluation. In *Proceedings of the 16th Pacific Asia Conference on Language, Information and Computation*, pages 69–78, 2001.

[76] Na-Rae Han and Shijong Ryu. Guidelines for Penn Korean Treebank Version 2.0. *IRCS Technical Reports Series*, page 7, 2005.

[77] Andrew Hardie. Developing a tagset for automated part-of-speech tagging in Urdu. In *Proceedings of the Corpus Linguistics Conference*, 2003.

[78] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[79] Sara Hussain. Finite-state morphological analyzer for Urdu. *Unpublished MS thesis, Center for Research in Urdu Language Processing, National University of Computer and Emerging Sciences, Pakistan*, 2004.

[80] Mark Johnson. PCFG Models of Linguistic Tree Representations. *Computational Linguistics*, 24(4):613–632. MIT Press., 1998.

[81] Dan Jurafsky. *Speech & Language Processing*. Pearson Education India, 2000.

[82] Kaarel Kaljurand. Checking treebank consistency to find annotation errors, 2004.

[83] Yoshihide Kato and Shigeki Matsubara. Parsing Gapping Constructions Based on Grammatical and Semantic Roles. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2747–2752, 2020.

[84] Yasuhiro Kawata and Julia Bartels. Stylebook for the Japanese Treebank in VERBMOBIL. In *Verbmobil-Report 240, Seminar für Sprachwissenschaft, Universität Tübingen*, 2000.

[85] Tafseer Ahmad Khan. Spatial expressions and case in South Asian languages. *Konstanzer Online-Publication-System (KOPS)*, 2009.

[86] Tafseer Ahmed Khan, Saba Urooj, Sarmad Hussain, Asad Mustafa, Rahila Parveen, Farah Adeeba, Annette Hautli, and Miriam Butt. The CLE Urdu POS tagset. In *LREC 2014, Ninth International Conference on Language Resources and Evaluation*, pages 2920–2925, 2015.

[87] Tafseer Ahmed Khan, Toqeer Ehsan, Almas Ashraf, Mutee U Rahman, Sarmad Hussain, and Miriam Butt. A Multilayered Urdu Treebank. In *7th International Conference on Language and Technology (CLT20)*, 2020.

[88] Paul Kingsbury, Martha Palmer, and Mitch Marcus. Adding Semantic Annotation to The Penn Treebank. In *Proceedings of the human language technology conference*, pages 252–256. San Diego, California, 2002.

[89] Eliyahu Kiperwasser and Yoav Goldberg. Simple and Accurate Dependency Parsing using Bidirectional LSTM Feature Representations. *Transactions of the Association for Computational Linguistics*, 4:313–327, 2016.

[90] Christo Kirov, Ryan Cotterell, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sebastian Mielke, Arya D McCarthy, Sandra Kübler, et al. UniMorph 2.0: Universal Morphology. *arXiv preprint arXiv:1810.11101*, 2018.

[91] Nikita Kitaev and Dan Klein. Constituency Parsing with a Self-Attentive Encoder. *arXiv preprint arXiv:1805.01052*, 2018.

[92] Nikita Kitaev and Dan Klein. Tetra-Tagging: Word-Synchronous Parsing with Linear-Time Inference. *arXiv preprint arXiv:1904.09745*, 2019.

[93] Dan Klein and Christopher D Manning. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.

[94] Dan Klein and Christopher D Manning. Fast Exact Inference with a Factored Model for Natural Language Parsing. In *Advances in Neural Information Processing Systems*, pages 3–10, 2003.

[95] Sandra Kübler. The PaGe 2008 Shared Task on Parsing German. In *Proceedings of the Workshop on Parsing German*, pages 55–63. Association for Computational Linguistics, 2008.

[96] Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 673–682. Association for Computational Linguistics, 2011.

[97] Yonggan Li, Xueguang Zhou, Yan Sun, and Huanguo Zhang. Design and implementation of Weibo sentiment analysis based on LDA and dependency parsing. *China Communications*, 13(11):91–105, 2016.

[98] Zhenghua Li, Min Zhang, Yue Zhang, Zhanyi Liu, Wenliang Chen, Hua Wu, and Haifeng Wang. Active learning for dependency parsing with partial annotation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 344–354, 2016.

[99] Jiangming Liu and Yue Zhang. In-Order Transition-Based Constituent Parsing. *Transactions of the Association for Computational Linguistics*, 5:413–424, 2017.

[100] Ting Liu, Jinshan Ma, and Sheng Li. Building a Dependency Treebank for Improving Chinese Parser. *Journal of Chinese Language and Computing*, 16(4): 207–224, 2006.

[101] Liangchen Luo. A Description of the CTB-to-Dependency Convertor. Technical report, Peking University, Beijing, 2018. URL `https://github.com/Luolc/CTB2Dep/blob/master/description.pdf`. Semester project report.

[102] Mohamed Maamouri and Ann Bies. Developing an Arabic Treebank: Methods, Guidelines, Procedures, and Tools. In *Proceedings of the Workshop on Computational Approaches to Arabic Script-based languages*, pages 2–9, 2004.

[103] Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. The Penn Arabic treebank: Building a large-scale annotated Arabic corpus. In *NEMLAR conference on Arabic language resources and tools*, volume 27, pages 466–467. Cairo, 2004.

[104] David M Magerman. Statistical Decision-Tree Models for Parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283. Association for Computational Linguistics, 1995.

[105] Muhammad Kamran Malik, Tafseer Ahmed, Sebastian Sulger, Tina Bögel, Atif Gulzar, Ghulam Raza, Sarmad Hussain, and Miriam Butt. Transliterating Urdu for a broad-coverage Urdu/Hindi LFG grammar. In *LREC 2010, Seventh International Conference on Language Resources and Evaluation*, pages 2921–2927, 2010.

[106] Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of the workshop on Human Language Technology*, pages 114–119. Association for Computational Linguistics, 1994.

[107] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[108] Yuval Marton, Nizar Habash, and Owen Rambow. Improving Arabic Dependency Parsing with Lexical and Inflectional Morphological Features. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 13–21, 2010.

[109] Mary L McHugh. Interrater reliability: The kappa statistic. *Biochemia medica: Biochemia medica*, 22(3):276–282, 2012.

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

[111] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[112] Tara Mohanan. *Argument Structure in Hindi*. Center for the Study of Language (CSLI), 1994.

[113] Simonetta Montemagni, Francesco Barsotti, Marco Battista, Nicoletta Calzolari, Ornella Corazzari, Antonio Zampolli, Francesca Fanciulli, Maria Massetani, Remo Raffaelli, Roberto Basili, et al. The Italian Syntactic-Semantic Treebank: Architecture, Annotation, Tools and Evaluation. 2003.

[114] Antonio Moreno, Ralph Grishman, Susana López, Fernando Sánchez, and Satoshi Sekine. A Treebank of Spanish and its Application to Parsing. In *LREC*, 2000.

[115] Phuong Thai Nguyen, Xuan Luong Vu, Thi Minh Huyen Nguyen, Hong Phuong Le, et al. Building a Large Syntactically-Annotated Corpus of Vietnamese. 2009.

[116] Phuong-Thai Nguyen, Anh-Cuong Le, Tu-Bao Ho, and Van-Hiep Nguyen. Vietnamese treebank construction and entropy-based error detection. *Language Resources and Evaluation*, 49(3):487–519, 2015.

[117] Quy T Nguyen, Yusuke Miyao, Ha TT Le, and Nhung TH Nguyen. Ensuring annotation consistency and accuracy for Vietnamese treebank. *Language Resources and Evaluation*, pages 1–47, 2017.

[118] Joakim Nivre, Mitchell Abrams, Ž Agić, et al. Universal Dependencies 2.4 (2019). *LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (UFAL), Faculty of Mathematics and Physics, Charles University*.

[119] Joakim Nivre, Jens Nilsson, and Johan Hall. Talbanken05: A Swedish Treebank with Phrase Structure and Dependency Annotation. In *LREC*, pages 1392–1395, 2006.

[120] Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. Maltparser: A Language-Independent System for Data-Driven Dependency Parsing. *Natural Language Engineering*, 13(2):95–135, 2007.

[121] Joakim Nivre, Željko Agić, Lars Ahrenberg, et al. Universal Dependencies 2.0. LINDAT/CLARIN digital library at the institute of Formal and Applied Linguistics, Charles University, Prague, 2017.

[122] Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. Universal Dependencies v2: An Evergrowing Multilingual Treebank Collection. *arXiv preprint arXiv:2004.10643*, 2020.

[123] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep Contextualized Word Representations. *arXiv preprint arXiv:1802.05365*, 2018.

[124] Slav Petrov and Dan Klein. Improved Inference for Unlexicalized Parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, 2007.

[125] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics, 2006.

[126] Ghulam Raza, Tafseer Ahmed, Miriam Butt, and Tracy Holloway King. Argument scrambling within Urdu NPs. *Proceedings of LFG11 Conference*, pages 461–481, 2011.

[127] Taneth Ruangrajitpakorn, Kanokorn Trakultaweekoon, and Thepchai Supnithi. A Syntactic Resource for Thai: CG Treebank. In *Proceedings of the 7th Workshop on Asian Language Resources (ALR7)*, pages 96–102, 2009.

[128] Hassan Sajjad. Statistical part of speech tagger for Urdu. *Unpublished MS Thesis, National University of Computer and Emerging Sciences, Lahore, Pakistan*, 2007.

[129] Hassan Sajjad and Helmut Schmid. Tagging urdu text with parts of speech: A tagger comparison. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 692–700. Association for Computational Linguistics, 2009.

[130] Federico Sangati and Willem Zuidema. Accurate parsing with compact tree-substitution grammars: Double-DOP. In *Proceedings of the conference on empirical methods in natural language processing*, pages 84–95. Association for Computational Linguistics, 2011.

[131] R Scha. Language Theory and Language Technology; Competence and Performance (in Dutch). In de Kort, Q. and Leerdam, G., editors. *Computertoepassingen in de Neerlandistiek*, 1990.

[132] Helmut Schmid. Treetagger — a language independent part-of-speech tagger. *Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart*, 43:28, 1995.

[133] Helmut Schmid. Improvements in part-of-speech tagging with an application to GERMAN. In *Natural language processing using very large corpora*, pages 13–25. Springer, 1999.

[134] Djamé Seddah, Grzegorz Chrupała, Özlem Çetinoğlu, Josef Van Genabith, and Marie Candito. Lemmatization and Lexicalized Statistical Parsing of Morphologically Rich Languages: The Case of French. In *Proceedings of the NAACL*

HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages, pages 85–93. Association for Computational Linguistics, 2010.

[135] Anthony Sigogne, Matthieu Constant, and Eric Laporte. French Parsing Enhanced with a Word Clustering Method Based on a Syntactic Lexicon. In *Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich Languages*, pages 22–27, 2011.

[136] Natalia Silveira, Timothy Dozat, Marie-Catherine De Marneffe, Samuel R Bowman, Miriam Connor, John Bauer, and Christopher D Manning. A Gold Standard Dependency Corpus for English. In *LREC 2014, Ninth International Conference on Language Resources and Evaluation*, pages 2897–2904, 2014.

[137] Khalil Sima'an, Alon Itai, Yoad Winter, Alon Altman, and Noa Nativ. Building a Tree-Bank of Modern Hebrew Text. *Traitement Automatique des Langues*, 42 (2):247–380, 2001.

[138] Gary F Simons and Charles D Fennig. Ethnologue: Languages of the world. *SIL International*, 20, 2017.

[139] Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. An Annotation Scheme for Free Word Order Languages. *arXiv preprint cmp-lg/9702004*, 1997.

[140] Richard Socher, Christopher D Manning, and Andrew Y Ng. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, volume 2010, pages 1–9, 2010.

[141] Richard Socher, John Bauer, Christopher D Manning, et al. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 455–465, 2013.

[142] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic

compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[143] Mitchell Stern, Jacob Andreas, and Dan Klein. A Minimal Span-Based Neural Constituency Parser. *arXiv preprint arXiv:1705.03919*, 2017.

[144] Juhi Tandon, Himani Chaudhry, Riyaz Ahmad Bhat, and Dipti Misra Sharma. Conversion from Paninian Karakas to Universal Dependencies for Hindi Dependency Treebank. In *Proceedings of the 10th Linguistic Annotation Workshop held in conjunction with ACL 2016 (LAW-X 2016)*, pages 141–150, 2016.

[145] Mariona Taulé, Maria Antònia Martí, and Marta Recasens. AnCora: Multilevel Annotated Corpora for Catalan and Spanish. In *Lrec*, 2008.

[146] Reut Tsarfaty and Khalil Sima'an. Three-Dimensional Parametrization for Parsing Morphologically Rich Languages. In *Proceedings of the Tenth International Conference on Parsing Technologies*, pages 156–167, 2007.

[147] Reut Tsarfaty, Djamé Seddah, Yoav Goldberg, Sandra Kübler, Marie Candito, Jennifer Foster, Yannick Versley, Ines Rehbein, and Lamia Tounsi. Statistical Parsing of Morphologically Rich Languages (SPMRL): What, How and Whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 1–12. Association for Computational Linguistics, 2010.

[148] Reut Tsarfaty, Djamé Seddah, Sandra Kübler, and Joakim Nivre. Parsing Morphologically Rich Languages: Introduction to the Special Issue. *Computational linguistics*, 39(1):15–22, 2013.

[149] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[150] Saba Urooj, Sarmad Hussain, Farah Adeeba, Farhat Jabeen, and Rahila Parveen. CLE Urdu digest corpus. *LANGUAGE & TECHNOLOGY*, 47, 2012.

[151] Xinyi Wang, Hieu Pham, Pengcheng Yin, and Graham Neubig. A tree-based decoder for neural machine translation. *arXiv preprint arXiv:1808.09374*, 2018.

[152] Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(2):207–238, 2005.

[153] Nianwen Xue and Martha Palmer. Annotating the Propositions in The Penn Chinese Treebank. In *Proceedings of the second SIGHAN workshop on Chinese language processing*, pages 47–54, 2003.

[154] Nianwen Xue, Fei Xia, Shizhe Huang, and Anthony Kroch. The Bracketing Guidelines for The Penn Chinese Treebank (3.0). *IRCS Technical Reports Series*, page 39, 2000.

[155] Daniel Zeman, Joakim Nivre, et al. Universal Dependencies 2.5. *Institute of Formal and Applied Linguistics, LINDAT/CLARIN, Charles University, Prague, Czech Republic, LINDAT/CLARIN PID: http://hdl. handle. net/11234/1-3105*, 2019.

[156] Junru Zhou, Zuchao Li, and Hai Zhao. Parsing all: Syntax and Semantics, Dependencies and Spans. *arXiv preprint arXiv:1908.11522*, 2019.

[157] Andreas Zollmann and Khalil Sima'an. A Consistent and Efficient Estimator for Data-Oriented Parsing. *Journal of Automata Languages and Combinatorics*, 10 (2/3):367, 2005.

APPENDIX A. DEPENDENCY STRUCTURE

A phrase structure (PS) treebank provides the constituency structure of the clauses and their hierarchical organization in a sentence. The information of the arguments is encoded at phrase level in the form of a linear order. On the other hand, the dependency structure (DS) focuses on the functional dependencies between lexical items of the clauses and dependency relations between constituents. It rules out the linear order and provides grammatical relations between predicates and their arguments.

The grammatical relations are crucial to obtain the information about event participants in the applications of natural language understanding (NLU). Phrase structure parsers usually provide the information of clauses in sentences which lack the semantic relations. However, functional labels can be attached with phrase labels to represent grammatical functions. On the other hand, the dependency treebanks and parsers provide the grammatical relations in the first place by ignoring the linear order of the constituents in contrast with phrase structure. Both types of treebanks exist for Urdu with various sizes and annotation schemes. The promising way forward is to convert existing phrase structure treebanks into a common dependency structure to train high-quality dependency parsers.

In this section, we present the conversion of our phrase structure treebank [55] into dependency structure by using Universal Dependencies label sets. To achieve an

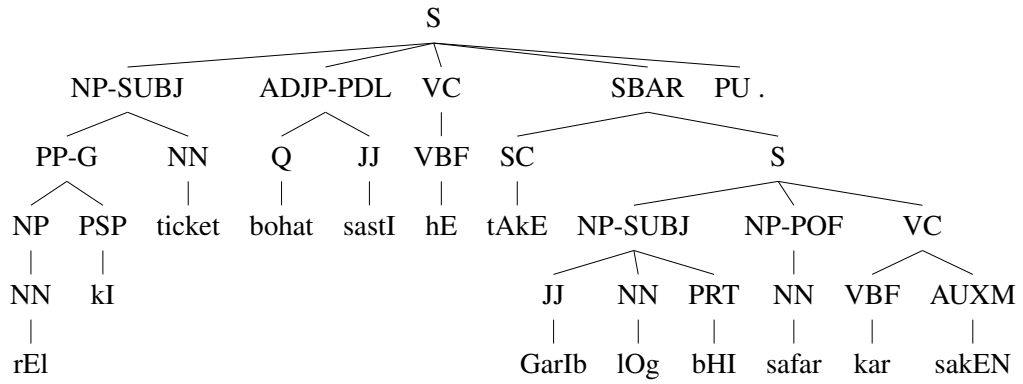
equivalent dependency structure, we devised a head word model for Urdu and a mapping from phrase to dependency labels. Additionally, several post-conversion rules were employed to achieve an accurate conversion.

The remainder of the chapter is organized as follows. Section A.1 briefly describes the compatibility of the phrase structure treebank with dependency structure. Section A.2 presents the conversion process by providing the details of head-word model, phrase to dependency label mapping and conversion rules. Section A.3 presents the treebank evaluation by showing the dependency parsing results.

A.1 COMPATIBILITY WITH DEPENDENCY STRUCTURE

We have developed a phrase structure treebank (CLE-UTB) which has a relatively flat annotation structure. The flat structure is helpful to annotate the flexible word order of the language. The annotation scheme has the ability to represent frequently observed syntactic constructions of the language including flexible word order, complex predicate structures, question phrases, subordinations, conjunctions, genitives and other cases, possessive phrases and relative clauses. Additionally, the treebank has a set of functional labels to mark grammatical relations. Figure A.1 presents a sample sentence from the CLE-UTB.

The PS parse tree in Figure A.1 shows the annotation of genitive case and a copula construction in the independent clause. The genitive case has been annotated by using a post-positional phrase (PP) and a functional label 'G'. The PP phrase is a component of a nominal subject which has *ticket* as the head. The independent clause further annotates a copula construction which marks an adjective phrase (ADJP) with a



rEl=kI	ticket	bohat	sastI	hE	tAkE	GarIb
train.F.Sg=Gen	ticket.F.Sg.Nom	very	cheap	be.Pres.3.Sg	SC	poor.Nom
lOg	bHI	safar	kar	sakEN	.	
people.Pl.Nom	also.PRT	travel.Nom	do	can.Pres.Mod.Pl	Punct	

'The train ticket is very cheap so that poor people can also travel'.

FIGURE A.1: A sample phrase structure parse tree.

functional label PDL (Predicate Link). The PDL construction makes a link with predicate which, in this case, is a copula verb *hE* 'be.Pres.3.Sg'.

The subordinate clause on the other hand contains a nominal subject and a complex predicate structure. The NP-SUBJ phrase also contains an intensifier *bHI* 'also'. The nominal subject follows a complex predicate structure which has been marked by using a POF (Part of Function) label. The noun phrase along with POF provides a verbal meaning with the VC (Verb Complex). In case of complex predicate structure, the predicate contains a light verb, *kar* 'do' in this case, which is usually followed by auxiliary verbs.

It is important to observe that all the arguments and predicates have been attached at the clausal level in the parse tree. This annotation allows the representation of sentences having different word orders. In the complex predicate structure, contributing noun, adjective or quantifier phrases usually have a pre-verbal order but this is not always the case. The functional layer makes this annotation flexible and complex

predicates can be marked irrespective of their appearance in the sentence. In the same way, copula constructions can be annotated independent of their position. Similarly, predicates and their arguments including subjects, objects, obliques, conjuncts, infinitive clauses, subordinate and relative clauses are annotated without restricting them to have specific positions. The annotation of the CLE-UTB makes it compatible with the dependency structure which represents the relations between lexical items and clauses. A detailed discussion of the compatibility of our PS annotation with DS is discussed in Section 3.2.2.

A.2 PHRASE TO DEPENDENCY CONVERSION

This section describes the Urdu head word model, phrase to dependency label mapping and post-conversion rules.

A.2.1 Head-word model

In the dependency structure, a head word is marked with the core dependency labels. A head word is the prominent lexical item which represents an argument. A head model finds the head words from the clauses by employing language dependent head rules [104]. We have proposed a model to identify heads in our PS treebank. For the conversion, we have updated an existing algorithm[1] [101] for the conversion. Table A.1 shows our head model.

Table A.1 presents word finding direction and label priority for each phrase label in the treebank. A verb complex usually contains a main verb followed by auxiliary

[1] https://github.com/Luolc/CTB2Dep

TABLE A.1: Head word model for Urdu treebank.

Phrase Label	Direction	Priority
VC	left	VBF, VBI, AUXA, AUXM, AUXP, AUXT, VC, NEG
PP	left	NP, S, QP, NNP, NN, PP, PSP
NP	right	NP, NNP, NN, PRP, PRR, S
ADJP	right	ADJP, JJ, Q, QP, RB
QP	right	QP, Q, CD, OD, FR, QM, JJ
ADVP	right	ADVP, RB, NP, NN
PREP	right	NP, NNP, NNP, PREP
DMP	right	PDM, PRP, PRT
FFP	left	FF, NNP, NN
S	left	VC, S, SBAR, NP, ADJP, QP NNP, NN, PRP,
SBAR	left	S, SBAR, SCK

verbs. Therefore, the head finding direction is from left to right with respect to label priorities. A VC normally has finite or infinitive verbs as main verbs, therefore, the POS tags VBF and VBI have high priorities. The verbal constituent in (13) shows a typical Urdu verbal structure which contains a main verb *likH* 'write', an aspectual auxiliary verb *liyA* 'take.Perf.M.Sg' and a tense auxiliary verb *hE* 'be.Pres.3.Sg'. The main verb should be the head to have the core dependency label which appears at left hand side. It is important to note that Urdu is written from write to left but its internal representation is a sequence of Unicodes. The directions in our head model have been devised for internal representations of the writing script.

(13) likH liyA hE

write.Imperf.Sg take.Perf.M.Sg be.Pres.3.Sg

'has written'

Similarly, the post-positional phrases have left heads as they contain at least one noun phrase which appears before a case marker. Therefore, the PP has NP to have highest priorities followed by other clause labels. In (14), a PP sequence contains a noun phrase *surx mEz* 'red table' followed by a locative case marker *par*. The head of the noun phrase is the head of the PP phrase. Therefore, the overall head direction of the PP is from left. Beside NPs, a PP can also contain 'S', QP etc., which have been added in its priority list.

(14) surx mEz=par

red.Sg table.M.Sg=Loc

'on the red table'

A noun phrase shown in (15) contains a genitive case hence an internal post-positional phase. The head of the NP is the word *pensil* 'pencil' which appears at the right hand side of the phrase. Similarly, the NP sequence presented in (16) shows a canonical NP structure which contains a head noun *Am* 'mango' after a noun modifier *mItHA* 'sweet'. The head direction of noun phrase in Urdu is from right. A noun phrase can also have internal NPs, common and proper nouns and pronouns which all are right headed.

(15) sAim=kI pensil

Saim.M.Sg=Gen pencil.F.Sg.Nom

'Saim's pencil'

(16) mItHA Am

sweet.Sg mango.M.Sg.Nom

'sweet mango'

The noun modifiers also have right heads as shown in (17) and (18). An adjective phrase, in the annotation, usually has quantifiers, cardinals, ordinals and main adjectives. The order of the adjective phrase is similar to NPs. The head modifier appears on the right hand side of the constituent. In the same way, the quantifier phrase also has right head as shown in (18).

(17) buhat mazbUt

very strong.M.Sg.Adj

'very strong'

(18) buhat kam

very less.M.Sg.Adj

'very less'

An adverbial phrase (ADVP) usually contains a single lexical entry for an adverb as presented in (19). In some cases, the ADVP contains infinitive clause in it but the head word remains on the right side of the constituent.

(19) acAnak

suddenly.Sg.Adv

'suddenly'

Urdu also has prepositional phrase but they are not frequent in the corpus. A prepositional phrase contains an NP following the preposition clitic making it right headed as shown in (20). The preposition clitic *fI* 'per' appears before a noun phrase making the noun as the head of the constituent.

(20) fI gHanTA

per hour.M.Sg.Nom

'per hour'

The demonstrative phrase (DMP) contains a demonstrative pronoun usually followed by a particle making its head finding method straight forward from right. Similarly, the FFP annotates the foreign fragment phrases which contain words from foreign languages usually from English and Arabic in our corpus. All the foreign words have been assigned a single POS tag i.e. FF. Therefore, we assume it left headed constituent. The 'S' label is used to annotate the clauses and a whole sentence. A sentence can have subordinate and coordinate clauses in it. The SBAR label has been used to annotate the subordinate clauses which further annotation at least one clause with label 'S'. The

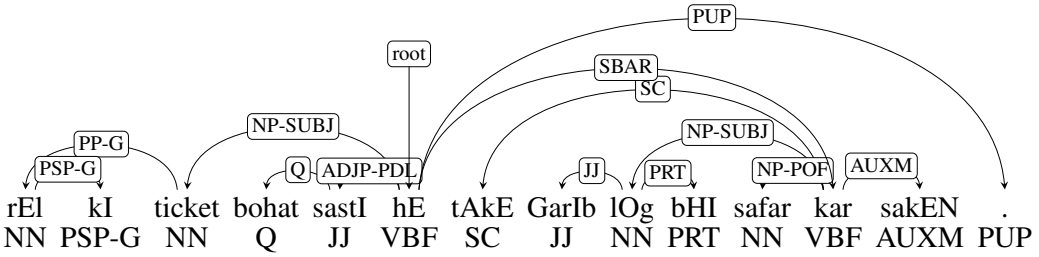


FIGURE A.2: An intermediate dependency representation from PS parse tree of Figure A.1 after head identification.

clause or sentence has the main verbal head as the head of the sentence. Therefore it finds the left head of the matrix clause as the head of the whole sentence. A hierarchical analysis and parental annotation has been performed to compute the heads for clause label 'S'. The parental annotation was helpful to attain the context of the phrases as the flat annotation does not provide much contextual information. The annotation was further helpful for the phrase to dependency label mappings.

Figure A.2 presents an intermediate dependency representation of the sentence from Figure A.1. The intermediate representation has been achieved by finding dependency arcs for the head words. The remaining items of the constituents show dependency arcs on the respective heads. The intermediate representation shows the phrase and functional labels as dependency relations. Beside the labels, the dependency arcs are quite correct. The sentence has the verb complex from independent clause as the root. The construction including subordinate clause, subjects, complex predicate, genitive case and auxiliary verbs have the dependency arcs on correct heads. A phrase to dependency label mapping is required to convert the intermediate representation to the dependency structure.

A.2.2 PS to DS Label Mappings

The intermediate dependency representation contains the dependency arcs for head and non-head tokens. The non-head words usually have POS tags as arc labels and heads show phrase labels for dependency relations. During the conversion, we have used updated version of the POS tag set which divides case markers and punctuations in different categories. The POS tag set of the CLE-UTB originally has 35 tags (Section 3.1.1) which were further extended to 44 tags (Section 4.3.1). The universal dependencies (UD) v-2.0 label sets have been used for dependency labels as well as POS tags. The universal POS tag set contains 17 tags [2]. The mapping from the CLE-UTB POS tag set to is presented in Table A.2. The mapping has been derived from universal dependencies guidelines[3] and the UD-2.0 version of the UDTB[4]. The POS mapping is quite straight-forward as more comprehensive tags are being mapped on abstract tags. For example, VERB tag is used for all verbs and AUX for all types of auxiliaries. Similarly, tags for ad-positions are mapped on a single ADP tag. Demonstrative, possessive, relative, reflexive apna, reflexive pronouns, ordinals, fractions and multiplicatives all are mapped on a single DET (determinant) tag. All punctuations are marked with a PUNCT tag. The tag set provides an abstraction which is applicable to many languages. At the second step, the phrase labels are mapped on dependency labels.

The intermediate dependency representation also shows the POS tags as arc labels as shown in Figure A.2. The noun *ticket* is a head word and shows a relation with label NP-SUBJ which is quite intuitive that it should be labeled as subject. The noun *rEl*

[2] https://universaldependencies.org/u/pos/
[3] https://universaldependencies.org/u/dep/
[4] https://github.com/UniversalDependencies/UD_Urdu-UDTB

TABLE A.2: The mapping of the CLE-UTB tags on UD-POS tags.

CLE-UTB POS	Universal POS	CLE-UTB POS	Universal POS
NN	NOUN	NEG	ADV
NNP	PROPN	PRE	ADP
VBI	VERB	PSP	ADP
VBF	VERB	PSP-E	ADP
AUXA	AUX	PSP-A	ADP
AUXP	AUX	PSP-SE	ADP
AUXT	AUX	PSP-I	ADP
AUXM	AUX	PSP-G	ADP
PRP	PRON	CC	CCONJ
PDM	DET	SC	SCONJ
PRS	DET	SCK	SCONJ
PRD	DET	SCP	SCONJ
PRR	DET	INJ	INTJ
PRF	DET	PRT	PART
APNA	DET	VALA	PART
JJ	ADJ	SYM	SYM
Q	ADJ	LRB	PUNCT
CD	NUM	RRB	PUNCT
OD	DET	PU	PUNCT
FR	DET	PU-C	PUNCT
QM	DET	PU-P	PUNCT
RB	ADV	PU-E	PUNCT

'train' shows a dependency on head with label PP-G. The label PP-G has been used to annotate post-positional phrases for genitive case markers. The Urdu script treats case marking clitics as independent tokens hence require a dependency label. The genitive case marker *kI* has a dependency on the noun *rEl* which is a possessor. The quantifier *bohat* 'very' has a dependency on the adjective *sastI* 'cheap' with a label Q which is a POS tag. The adjective *sastI* is the head word with a core dependency label ADJP-PDL. It makes a predicate link with copula verb *hE*. Such copula constructions have

been updated by employing additional rules after the label mapping.

On the other hand, the subordinate clause contains a subject and a complex predicate structure. The phrase labels, NP-SUBJ and NP-POF represent the dependency relations of heads and POS tags have been labeled to show non-core relations. For example, the noun modifier *GarIb* 'poor' has a dependency on head noun *lOg* 'people' with the label JJ. The modal auxiliary is defined by the tag AUXM. The particle *bHI* 'also' has the label PRT. The subordinate clause is marked with the SBAR label. After the analysis, it is quite intuitive that phrase label along with functional labels represent the dependency relations of heads on the predicates and the POS tags mark the relations of non-head items on the constituent heads. Therefore to achieve dependency labels, the mapping on dependency labels should be against phrase labels as well as POS tags.

To achieve the context of the constituents for accurate label mapping, we updated the CLE-UTB to have parental annotations [80]. A clause 'S' has different attachments which can be identified by using parent label annotation along with the actual label. For example, if the clause 'S' appears under a noun phrase, it is annotated as SˆNP and if it is attached with a post-positional phrase, it would have the annotation as SˆPP. The subordinate and coordinate clauses are also annotated with an S label which are resulted to show SˆSBAR for subordinate clause and SˆS for a coordinate clause. Addition to the parent label annotation, tree levels were also annotated with VCs which were helpful to identify roots of sentences. A sentence with many clauses usually has many verbal constructions in the hierarchical parse trees. The level number annotation attached a number, for example, VCˆ1 represents the verbal construction at first level if we consider sentence clause at zeroth level.

TABLE A.3: The mapping of the CLE-UTB phrase labels on UD labels for core arguments.

CLE-UTB Lables	UD-2.0 Label	UD Description
NP-SUBJ	nsubj	Nominal subject
PP-SUBJ	nsubj	Nominal subject
QP-SUBJ	nsubj	Nominal subject
NP-OBJ	obj	Object
PP-OBJ	obj	Object
QP-OBJ	obj	Object
S-OBJ	obj	Object
NP-OBL*	iobj	Indirect subject
S-SUBJ	csubj	Clausal subject
SˆS	ccomp	Clausal complement
SBAR	ccomp	Clausal complement
SˆSBAR	ccomp	Clausal complement
S*	xcomp	Open clausal complement

∗The dependency labels were derived by additional rules after conversion.

A.2.2.1 Core Arguments

Table A.3 shows the mapping of the phrase labels of core arguments to the UD labels. The functional labels represent core arguments for subject and object. The subordinate and coordinate clauses are labels with `ccomp` (clausal complement) label. The parental annotations have been used to identify the clauses. The nominal obliques have been mapped on indirect objects `iobj`. Infinitive clausal objects have been marked as indirect objects with label `xcomp`. The labels with '*' have been achieved by additional rules described in Section A.2.3.

A.2.2.2 Non-Core Dependents

Table A.4 presents the mapping of non-core dependencies. The non-core compulsory arguments have been annotated by using OBL functional label which is mapped on oblique dependency with `obl` label. The functional labels VOC has been mapped on `vocative` label. Non-core finite clauses have been marked by using `advel` (adverbial clause modifier) label. Similarly, clauses with conjunctive participle are also marked with `advel`. These types of clauses are attached under SBAR label which have been identified by using parental annotation. Adverbial phrases (ADVP) and nominal adjuncts have been mapped on adverbial modifier dependency by using label `advmod`. The noun phrase, post-positional phrase, prepositional phrase and quantifier phrase without a functional label have been considered as adjuncts and are marked with adverbial modifier. The POS tags for adverb RB and negative adverb NEG are mapped on the `advmod` label.

The adverbial interjections have been marked with the `discourse` label. All types of auxiliary verbs and conjunctive participles are mapped on `aux` label. The annotation of the CLE-UTB uses a PDL functional label to mark predicate link with copula verbs. The constructions with PDL labels are mapped on copula dependency with `cop` label. The labels for subordinate clitics are mapped on `mark` dependency label.

A.2.2.3 Nominal Dependents

Table A.5 shows the mapping of nominal dependents. The genitive cases are marked as noun specifiers and possessors. The genitive case has been annotated with

TABLE A.4: The mapping of the CLE-UTB phrase labels on UD labels for non-core dependents.

CLE-UTB Lables	UD-2.0 Label	UD Description
NP-OBL	obl	Oblique nominal
PP-OBL	obl	Oblique nominal
S-OBL	obl	Oblique nominal
ADVP-VOC	vocative	Vocative
NP-VOC	vocative	Vocative
S	advcl	Adverbial clause modifier
S^SBARSCK	advcl	Adverbial clause modifier
SBARSCK	advcl	Adverbial clause modifier
ADVP	advmod	Adverbial modifier
NP	advmod	Adverbial modifier
NP-ADJ	advmod	Adverbial modifier
PP	advmod	Adverbial modifier
PREP	advmod	Adverbial modifier
QP	advmod	Adverbial modifier
NEG	advmod	Adverbial modifier
RB	advmod	Adverbial modifier
ADVP-INJ	discourse	Discourse element
INJ	discourse	Discourse element
AUXA	aux	Auxiliary
AUXM	aux	Auxiliary
AUXP	aux	Auxiliary
AUXT	aux	Auxiliary
SCK	aux	Auxiliary
ADJP-PDL	cop	Copula
ADVP-PDL	cop	Copula
NP-PDL	cop	Copula
PP-PDL	cop	Copula
QP-PDL	cop	Copula
SC	mark	Marker
SCP	mark	Marker

PP-G phrase label in the phrase structure. For conversion from the phrase to dependency structure, it is marked as nominal modifier using the label `nmod`. Cardinals are marked as numeric modifier by using the label `nummod`. The adjectival phrases and adjectival tags are mapped as adjectival modifiers. The adjectival clauses which have been identified by using parental annotation for clause label 'S', are mapped on `acl` label. The determiner label `det` has been used to represent demonstrative phrase, APNA particle, fractions, ordinals, quantifiers and multiplicatives. Different types of pronouns including demonstrative, relative demonstrative, reflexive, relative and possessive pronouns are marked as determiners. The dependency label `case` has been used to map the annotation of all types of case markers.

A.2.2.4 Other Dependency Relations

Table A.6 presents the dependency labeling for coordinations, multi-word expressions, punctuations, roots and other unspecified dependencies. The conjunctions within noun, adjective and quantifier phrases annotated with commas, are identified on the bases of POS tags and are marked with `conj` label. Coordinate conjunction tag CC has been marked with dependency label `cc`. The noun, adjective and quantifier phrases which have been associated by using an internal case marker PSP-I, are marked by using `fixed` label. The foreign fragment phrases usually have flat structure in them. Therefore they are marked as `flat` in the dependency structure. The compound nouns with more than one lexical items have right word as head and the non-head items have been marked with `flat` label. The example of such noun phrase is *mOm battI* 'candle'. In Urdu, both *mOm* 'wax' and *battI* 'light' are nouns. In the head model, the noun *battI* is

TABLE A.5: The mapping of the CLE-UTB phrase labels on UD labels for nominal dependents.

CLE-UTB Lables	UD-2.0 Label	UD Description
PP-G	nmod	Nominal modifier
CD	nummod	Numeric modifier
ADJP	amod	Adjectival modifier
JJ	amod	Adjectival modifier
SˆADJP	acl	Adjectival clause
SˆNP	acl	Adjectival clause
SˆPP	acl	Adjectival clause
DMP	det	Determiner
APNA	det	Determiner
FR	det	Determiner
OD	det	Determiner
PDM	det	Determiner
PRD	det	Determiner
PRF	det	Determiner
PRP	det	Determiner
PRR	det	Determiner
PRS	det	Determiner
Q	det	Determiner
QM	det	Determiner
PRE	case	Case marking
PSP	case	Case marking
PSP-G	case	Case marking
PSP-I	case	Case marking
PSP-A	case	Case marking
PSP-E	case	Case marking
PSP-SE	case	Case marking

the head and *mOm* has a flat dependency relation on the head.

The annotation of complex predicate structures is represented by using the POF functional label. The functional label is usually attached with noun, adjective and quantifier phrases. In the label mapping, the phrases with POF label have been mapped on

TABLE A.6: The mapping of the CLE-UTB phrase labels on UD labels for dependency relations.

CLE-UTB Lables	UD-2.0 Label	UD Description
NP*	conj	Conjunct
NN*	conj	Conjunct
ADJP*	conj	Conjunct
JJ*	conj	Conjunct
QP*	conj	Conjunct
Q*	conj	Conjunct
CC	cc	Coordinating conjunction
NP*	fixed	Fixed multiword expression
ADJP*	fixed	Fixed multiword expression
QP*	fixed	Fixed multiword expression
FF	flat	Flat multiword expression
NN	flat	Flat multiword expression
NP-POF	compound	Compound
ADJP-POF	compound	Compound
QP-POF	compound	Compound
VC-VALA	compound	Compound
LRB	punct	Punctuation
PU	punct	Punctuation
PU-C	punct	Punctuation
PU-E	punct	Punctuation
PU-P	punct	Punctuation
RRB	punct	Punctuation
VC	root	Root
FFP	dep	Unspecified dependency
PRT	dep	Unspecified dependency

∗The dependency labels were derived by additional rules after conversion.

`compound` label. The verbal construction with *vAlA* particles are also marked as compounds. All types of punctuations have been mapped on a single `punct` label. The head of the verbal construction has been annotated as `root` of the sentence. The verbal structure of the independent clause at the highest level in tree hierarchy has been

marked as root. The label `dep` is used to mark unspecified dependencies. The items on the foreign fragment phrases have flat dependencies on head but its head also has an arc to the root of the sentence. We have marked such relations as unspecified dependencies. Similarly, the particles are used as intensifiers hence marked with `dep` label in the conversion process.

A.2.3 Post Conversion Rules

For many phrase labels, the dependency mapping is quite straight forward as the phrase annotation scheme has functional labels to mark grammatical relations. Due to flat annotation of the CLE-UTB, complex structures were marked by using post-conversion rules to achieve an accurate dependency treebank. The first main issue was with indirect objects. The annotation of the CLE-UTB does not annotate secondary objects but rather it marks them as obliques by attaching OBL labels. However, Urdu uses accusative case marker *kO* for secondary objects and recipients. Similarly, the non-finite clauses were annotated just like other finite clauses. These clauses were identified from their verbal heads. Further rules were derived for `fixed` and `conj` label. The details of conversion rules are as follows.

- One way to denote secondary objects and recipients is the use of accusative/dative case marker *kO* in Urdu. The annotation of the CLE-UTB labels all case marking constructions with post-positional phrase (PP). The annotation marks secondary objects as obliques hence with the label PP-OBL. If the label is PP-OBL and the next lexical item is the clitic *kO* then mark it with `iobj` dependency label.

- There are also nominal secondary objects which are also annotated as nominal obliques. These constructions normally have special types of pronouns which provide the meaning of recipients or beneficiaries. A list of such pronouns with their meanings is shown in Table A.7. If the label is NP-OBL and the head of the phrase is any of these pronouns then mark the dependency arc with label `iobj`.

TABLE A.7: List of pronouns marked as indirect objects.

Pronouns	Meaning
mujHE	To me
hamEN	To us
tujHE	To you
tumEN	To you
isE	To him/her
usE	To him/her
inhEN	To them
unhEN	To them
jisE	To whom (Sg)
jinhEN	To whom (Pl)
kisE	To whom

- The CLE-UTB annotates the non-finite clauses and clausal objects similar to other clauses by using labels 'S' and S-OBJ. We have mapped such constructions on the `xcomp` dependency label. If a construction has any of these labels and their phrasal head is an infinitive verb having the VBI POS tag then mark the construction as `xcomp` in the dependency structure.

- In the phrase structure annotation of the CLE-UTB, some constructions contain an internal clitic as a connection between two nouns, adjectives and quantifiers. For instance the constituent *kam-az-kam* 'at least' has a clitic *az* 'from'. The clitic

has been tagged with the POS tag PSP-I. Such constructions with tag PSP-I have been marked with a `fixed` dependency label.

- The conjunctions between nouns, adjectives and quantifiers are represented by using conjunction clitics and comma. The comma is used when there are more than two items. The label `conj` has been marked for such constructions if they use conjunctions with the CC POS tag or comma with PU-C tag.

Figure A.3 shows the final dependency tree from the intermediate structure after applying the label mappings and conversion rules.

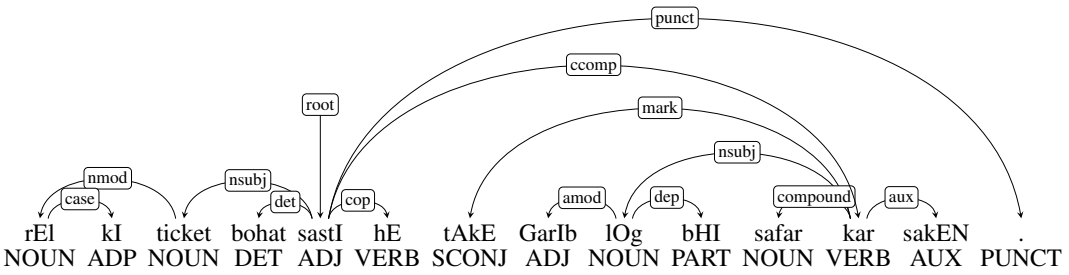


FIGURE A.3: Final dependency tree from the intermediate tree of Figure A.2 after label mapping and conversion rules.

The dependency arcs in Figure A.3 are quite similar to the arcs of Figure A.2 except copula construction. All phrase labels have been replaced by the dependency labels. The copula construction has been updated with respect to dependency labels. The predicate link has been marked as `root` of the sentence and copula verb *hE* shows a `cop` dependency on the root. The conversion rules were helpful to achieve accurate mapping for `iobj`, `xcomp`, `fixed` and `conj` labels. The converted treebank has 28 unique dependency labels which are shown in Table A.8.

TABLE A.8: UD labels which have been used by converted dependency treebank.

Universal dependency labels			
acl	advcl	advmod	amod
aux	case	cc	ccomp
compound	conj	cop	csubj
dep	det	discourse	fixed
flat	iobj	mark	nmod
nsubj	nummod	obj	obl
punct	root	vocative	xcomp

A.3 DEPENDENCY PARSING

For the evaluation of the converted dependency treebank, we have trained the well-known MaltParser [120]. It is a data-driven parsing system which is based on arc-eager transition algorithm. The arc-eager algorithm produces better parsing results with better efficiency as compared to arc-standard algorithm [32]. Default parameters have been used to train the MaltParser including the POS tags along with lexical items.

A transition-based BiLSTM (Bi-directional long-short term memory) dependency parser[5] [89] was further trained to achieve state of the art dependency parsing results for newly converted treebank. The parser computed the embeddings for tokens and POS tags which were further combined to achieve a single vector for each token. A multi-layer perceptron (MLP) was used to score feature vectors with one hidden layer. The BiLSTM is best known for its learning of sequential labels like dependency labels. The parser further used an arc-hybrid system [96] for the prediction of dependencies. The arc-hybrid system uses an efficient dynamic oracle as described in [64]. The model performs three transition tasks which are $SHIFT$, $LEFT_{label}$ and $RIGHT_{label}$.

[5]https://github.com/elikip/bist-parser

The $SHIFT$ operations transfer the first entry from the input buffer onto the stack, the $LEFT_{label}$ operation gets the top element of the stack and makes it a modifier to the first element of the input buffer and the $RIGHT_{label}$ operation gets the top element of the stack, makes it a modifier and attaches it to the current top element of the stack.

The training configurations and parameters of the BiLSTM parser are; two LSTM hidden layers, 125 hidden LSTM dimensions, MLP hidden dimensions of 100, *tanh* activation, dropout rate of 0.25, and adam optimizer. These configurations have been used for all experiments of the BiLSTM parser. The parser was trained for 20 epochs and produced a trained model after every epoch. However, the best model was selected on the basis of labeled attachment score (LAS) on the development set. The train set contained 6,135 sentences with a development set of 746 sentences. The transfer leaning was further used to improve the results by computing word embeddings for a large unannotated Urdu corpus. The BiLSTM parser performed with a best labeled attachment score of 84.2, an unlabeled attachment score (UAS) of 89.6 and a label accuracy (LA) of 90.3. Table A.9 presents the parsing results for the MaltParser and the BiLSTM parser.

The BiLSTM dependency parser outperforms the MaltParser on the converted dependency Urdu treebank (CLE-UDTB). Both parsers have been trained by using POS tags as syntactic feature along with lexical elements. The transfer learning has been performed by training Word2vec [111] and ELMo [123] word embeddings. A plain Urdu corpus has been used for training the embeddings which contains 35 million words. The embedding vocabulary contained 72 thousand words. The embedding dimensions were 100 and 128 for Word2Vec and ELMo respectively. These word embeddings were

TABLE A.9: Dependency parsing results for the newly converted Urdu DS treebank.

Results by using gold POS tags				
Parser	Emb.	UAS	LAS	LA
MaltParser	-	88.3	81.6	88.5
BiLSTM Parser	-	89.1	83.3	89.8
	W2V	89.3	83.7	90.1
	ELMo	89.6	84.2	90.3
Results by using predicted POS tags with an accuracy of 96.3%				
Parser	Emb.	UAS	LAS	LA
MaltParser	-	85.3	78.2	86.3
BiLSTM Parser	-	86.3	80.1	87.6
	W2V	86.3	80.3	87.9
	ELMo	87.1	81.2	88.4

helpful to improve the parsing results of the BiLSTM dependency parser.

The improvements of the dependency parsing for Hindi and Urdu treebanks of HUTB have been presented in [13]. They have used syntactically rich features in their experiments. Their baseline model used POS tags, chunk tags, root forms of the words and cluster IDs of words as basic features. The baseline model achieved LAS of 81.19, UAS of 88.77 and LA of 84.84 for the Urdu dependency treebank. To improve the dependency results, additional features were incorporated which included agreements, cases, complex predicates and *ezafe* constructions [21]. By using these features, the best scores achieved were LAS of 83.21, UAS of 90.39 and LA of 86.92. Our treebank, on the other hand, contains POS information only as the morpho-syntactic feature. However the MaltParser, based on the arc-eager algorithm, produces comparative dependency results for both treebanks. The BiLSTM parser produces the improved results by learning hidden dependencies by transfer learning. We developed a POS tagger which is also based on BiLSTM networks. The POS tagger performed the tagging with

an accuracy of 96.3%. Both the MaltParser and the BiLSTM parser produced high label accuracies for all our experiments. On the basis of the conversion process and dependency results, it is concluded that the phrase structure of the CLE-UTB is compatible with the dependency structure. We will further evaluate the treebank conversion by preparing a manually annotated reference corpus.

APPENDIX B. LABEL SETS

B.1 HUTB LABEL MAPPING ON CLE-UTB LABELS

TABLE B.1: Comparison of the CLE-UTB functional labels with the HUTB dependency labels.

Sr#	Dependency label (HUTB)	Description	Category	Func. label (CLE-UTB)
1	k1	Karta	Doer/Subject/Agent	SUBJ
2	pk1	Prayojaka Karta	Causer	SUBJ
3	k4a	Anubhava Karta	Experiencer	SUBJ
4	k2	Karma	Object/Patient	OBJ
5	k4	Sampradana	Recipient	OBL
6	k2p		Destination/Goal	OBL
7	jk1	Prayojya Karta	Causee	OBL
8	mk1	Madhyastha Karta	Mediator causee	OBL
9	k2g	Secondary Karma	Secondary object	OBL
10	k2s	Karma Samanadhikarana	Object complement	OBL
11	k3	Karana	Instrument	OBL
12	k5prk	Prakruti apadana	Source material	OBL
13	k5	Apadana	Source	OBL
14	Rd	Relation Prati	Direction	ADJ
15	k7t	KAlAdhikarana	Location in time	ADJ
16	k7p	Deshadhikarana	Location in space	ADJ
17	k7	Vishayadhikarana	Location elsewhere	ADJ
18	ras_k∗	Upapada sahakArakatwa	Associative	ADJ
19	ras_NEG		Negation in associative	ADJ
20	rt	Tadarthya	Purpose	ADJ
21	rh	Hetu	Reason	ADJ
22	k∗u	Sadrishya	Similarity	ADJ
23	r6_k1	Karta of a conjunct verb	Complex predicate	POF
24	r6_k2	Karma of a conjunct verb	Complex predicate	POF
25	pof	Part of function	Complex predicate	POF
26	r6v		Relation between noun & verb	PDL
27	k1s	Karta samanadhikarana	Noun complement of karta	PDL
28	rad		Address terms	VOC
29	r6	Shashthi	Genitive/Possessive	G

VITA

Mr. Toqeer Ehsan is currently pursuing his Ph.D in Computer Science in the domain of Natural Language Processing (NLP). He has been serving University of Gujrat as a Lecturer since 2010. Before joining UOG, he worked in the software industry for three years. His research interests are Machine Learning, Deep Learning, Computational Linguistics and Natural Language Processing. His research focus is on creating computational resources for Pakistani Languages like Urdu and Punjabi (Shahmukhi). He has published several research papers in the aforementioned domains. He also supervises research students in the domain of Computational Linguistics (CL), NLP and Text processing.