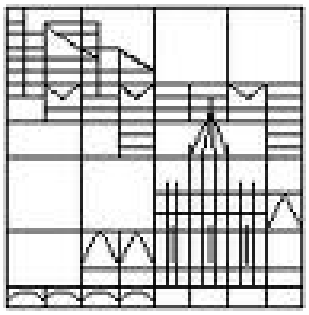
Building Computational Resources: The URDU.KON-TB Treebank and the Urdu Parser

Dissertation zur Erlangung des akademischen Grades eines Doktors der Philosophie

vorgelegt von

Qaiser Abbas
Fachbereich Sprachwissenschaft
Universität Konstanz



Referent (Chair): Prof. Dr. Oliver Deussen
Referentin: Prof. Dr. Miriam Butt
Referentin: Prof. Dr. Heike Zinsmeister

Tag der mündlichen Prüfung: 17 September, 2014

I certify that I have read this dissertation comprises computational linguistics work and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Prof. Dr. Oliver Deussen,
Computer and Information Science, Universität Konstanz.

I certify that I have read this dissertation comprises computational linguistics work and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Prof. Dr. Miriam Butt,
General and Computational Linguistics, Universität Konstanz.

I certify that I have read this dissertation comprises computational linguistics work and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Prof. Dr. Heike Zinsmeister,
General and Corpus Linguistics, Universität Hamburg.

Signature from the head of PhD committee

I dedicate this dissertation to my wife, Sadia Niazi and to my kids,
Minahil Qaiser and Ali Azmi.

Acknowledgements

First of all, I owe it all to Almighty God for granting me the wisdom, health and strength to undertake this research task and enabling me to its completion.

Completion of this doctoral dissertation was possible with the support of several people. I would like to express my sincere gratitude to all of them. First of all, I am extremely grateful to my doctoral supervisor, Prof. Miriam Butt, for her valuable guidance, scholarly inputs and consistent encouragement I received throughout my doctoral studies. This feat was possible only because of the unconditional support provided by her. A person with an amicable and positive disposition, Prof. Butt has always made herself available to clarify my doubts despite her busy schedules and I consider it as a great opportunity to do my doctoral programme under her guidance and to learn from her research expertise. Thank you Prof. Butt, for all your help and support. I thank Prof. Oliver Deussen and Prof. Heike Zinsmeister, for their reviewing support and the guidance provided to carry out my research work. They have been very encouraging and supportive, and I express my gratitude to them.

The dissertation would not have come to a successful completion, without the help I received from my colleagues. I would like to thank Dr. Ghulam Raza and Dr. Tafseer Ahmed, for their guidance services in connection with Urdu and special thanks to Christin Schätzle, who translated my dissertation abstract into German. I found my other colleagues including Tina Bögel, Sebastian Sulger, Annette Hautli, Jaouad Mousser, Achim Kleinmann, etc, always very much kind and patient. They always willing to lend their services whenever I approached them. I acknowledge and appreciate them for all their support and efforts.

I owe a lot to my parents, who encouraged and helped me at every stage of my personal and academic life, and one of them longed to see this achievement come true. I deeply miss my mother Faiz Naseem, who is no more in this world to share this joy with me. May Allah grant her peace and admit her in Jannah. My father, Khaliq Dad Khan, being an old man, has been a custodian for all my house related matters during my doctoral studies. He performed just like a great father again even in this very old age. He supported me in every possible way to see the completion of this work.

I thank my wife, Sadia Niazi and my brothers, Ghulam Ali Asghar, Imran Raza and Khawar Abbas, for their wonderful services all these years. Specially, my wife, she has been a great support ever, and I owe her so much for her care and hospitality. I also thank my sisters and their families, father and mother in law, brothers-in-law and their families, for their good wishes. Lastly, my great wishes and love for my kids, Minahil Qaiser and Ali Azmi. May Allah keep them always in peace and good health.

Abstract

This work presents the development of the URDU.KON-TB treebank, its annotation evaluation & guidelines and the construction of the Urdu parser for a South Asian language Urdu. Urdu is comparatively an under-resourced language and the development of a reliable treebank and a parser will have significant impact on the state-of-the-art for automatic Urdu language processing.

The work includes the construction of the raw corpus containing 1400 sentences collected from Urdu Wikipedia and the Jang newspaper. The corpus contains text of local & international news, social stories, sports, culture, finance, religion, traveling, etc. The hierarchal annotation scheme adopted has a combination of phrase structure and hyper dependency structure. A semi-semantic part of speech tag set, a semi-semantic syntactic tag set and a functional tag set are proposed, which are further revised during the annotation of the raw corpus. The annotation of the sentences was performed manually. Due to the addition of morphology, part of speech, syntactical, semantical, clausal, grammatical and miscellaneous features, the annotation scheme is linguistically rich. The annotation resulted in a treebank for Urdu, called the URDU.KON-TB. This is presented in Chapter 3.

For an evaluation of the annotation scheme, Krippendorff's α co-efficient is selected. This is a statistical measure to evaluate inter-annotator agreement. Randomly selected 100 sentences from the URDU.KON-TB treebank were given to five trained annotators for annotation. The annotated sentences then evaluated using the Krippendorff's α co-efficient. The α values of inter-annotator agreement obtained for part of speech, syntactical and functional annotation are 0.964, 0.817 and 0.806, respectively. The evaluation is presented in Chapter 4. All of the three values lie in the range of

perfect agreement. The annotation guidelines devised in the development of the URDU.KON-TB treebank were revised during and after this annotation evaluation. The updated version is presented in Chapter 2.

For the development of an Urdu parser, 1400 annotated sentences in the URDU.KON-TB treebank are divided into 80% training data and 20% test data. A context free grammar is extracted from this training data, which is then given to the Urdu parser after its development. The test data is divided into 10% held out data and 10% test data. The test data then contains 140 sentences with an average length of 13.73 words per sentence. The held out data is used during the development of the Urdu parser. Urdu parser is an extended version of dynamic programming algorithm known as the Earley parsing algorithm. The extensions made are discussed in Chapter 5 along with the issues faced during the development. All items which can occur in a normal text are considered, e.g., punctuation, null elements, diacritics, headings, regard titles, Hadees (the statements of prophets), anaphora with in a sentence, and others. The PARSEVAL measures are used to evaluate the results of the Urdu parser. By applying a sufficiently rich grammar along with the extended parsing model, the parser gives 87% of f-score and outperforms the multi-path-shift-reduce parser for Urdu, a two stage Hindi dependency parser and a simple Hindi dependency parser with 4.8%, 12.48% and 22% increase in recall, respectively.

The URDU.KON-TB treebank and the Urdu parser is a contribution to the overall computational resources of Urdu. By-products of this work are a semi-semantic part of speech tagset, a semi-semantic syntactic tagset, a functional tagset, annotation guidelines, a grammar with sufficient encoded information for parsing of morphologically rich language Urdu and a part of speech tagged corpus, which can be used for the training of part of speech taggers. These resources will be enhanced further and can be used for natural language processing such as probabilistic parsing, training of POS taggers, disambiguation of spoken sentences, grammar development, language identification, sources for linguistic inquiry and psychological modeling, or pattern matching.

Zusammenfassung

Die vorliegende Arbeit zeigt die Entwicklung der URDU.KON-TB Treebank sowie die Evaluation und Richtlinien ihrer Annotation und die Konstruktion des Urdu-Parsers für die südasiatische Sprache Urdu. Für Urdu gibt es vergleichbar wenig sprachliche Ressourcen, weshalb die Entwicklung einer verlässlichen Baumbank und eines Parsers einen signifikanten Einfluss auf den aktuellen Stand der automatischen Sprachverarbeitung Urdu haben wird.

Des Weiteren wird ein Rohkorpus bestehend aus 1400 Sätzen konstruiert, die aus Urdu Wikipedia und der Tageszeitung Jang stammen. Der Korpus beinhaltet somit Texte mit verschiedenen Thematiken: Lokale und internationale Nachrichten, Sozialgeschichte, Sport, Kultur, Finanzwirtschaft, Religion, Reisen und vieles mehr. Das hierarchische Annotationsschema des Korpus ist eine Kombination aus Phrasenstruktur und Hyperdependenzstruktur. Es werden ein semi-semantisches Part-of-speech Tagset, ein semi-semantisches syntaktisches Tagset sowie ein funktionales Tagset vorgeschlagen, die während der Annotation des Rohkorpus überarbeitet wurden. Die Annotation der Sätze wurde manuell vorgenommen. Aufgrund der Hinzufügung von Morphologie, Part-of-speech, syntaktischen, semantischen, klausalen, grammatischen und weiteren Eigenschaften, ist das Annotationsschema linguistisch stark angereichert. Das Ergebnis der Annotation ist die URDU.KON-TB Treebank, eine Treebank für Urdu. Diese wird in Kapitel 3 präsentiert.

Zur Evaluation des Annotationsschemas dient Krippendorffs α-Koeffizient. Dieser Koeffizient ist ein statistischer Messwert zur Gütebestimmung des Inter-annotator Agreements. Fünf trainierte Annotierer nahmen die Annotation von 100 zufällig ausgewählten Sätzen aus der URDU.KON-TB

Treebank vor. Anschlieend wurden die annotierten Sätze durch den Krippendorffs α-Koeffizienten evaluiert. Die α des Inter-annotator Agreements für Part-of-speech, syntaktische und funktionale Annotation liegen jeweils bei 0.964, 0.817 und 0.806. Diese Werte liegen im Bereich der perfekten Übereinstimmung. Die Evaluation wird in Kapitel 4 behandelt. Die Annotationsrichtlinien, die während der Entwicklung der URDU.KON-TB Treebank entworfen wurden, wurden innerhalb und nach dieser Annotationsauswertung überarbeitet. Eine aktualisierte Version wird in Kapitel 2 vorgestellt.

Für die Entwicklung eines Urdu-Parsers wurden 1400 annotierte Sätze aus der URDU.KON-TB Treebank in 80% Trainings- und 20% Testdaten aufgeteilt. Aus den Trainingsdaten wurde eine kontextfreie Grammatik extrahiert, die anschlieend an den entwickelten Urdu-Parser weitergegeben wurde. Die Testdaten bestehen aus 140 Sätzen mit einer durchschnittlichen Länge von 13,73 Wörtern pro Satz. Diese Daten wurden während der Entwicklung des Urdu-Parsers benutzt. Der Urdu-Parser ist eine erweiterte Version des sogenannten Earley-Algorithmus, ein dynamischer Programmieralgorithmus. Die erforderlichen Erweiterungen sowie die Probleme, die während der Entwicklung auftraten, werden in Kapitel 5 diskutiert. Alle typischen Merkmale eines normalen Textes wurden miteinbezogen, z.B. Interpunktion, Null-Elemente, Diakritika, Überschriften, Hadees (Aussagen von Propheten), Anaphora innerhalb eines Satzes und weitere. Die PARSEVAL-Mastäbe werden zur Evaluation des Urdu-Parsers genutzt. Dank der reichen Grammatik und dem erweiterten Parsingmodell gibt der Parser einen 87%-igen F-score und überholt somit den Multi-path-shift-reduce Parser für Urdu, einen zweiteiligen Hindi-Dependenzparser und einen einfachen Hindi-Dependenzparser mit jeweils 4,2%, 12,48% und 22% Recall.

Die URDU.KON-TB Treebank und der Urdu-Parser sind ein Beitrag zu den gesamten computerlinguistischen Ressourcen für Urdu. Nebenprodukte der vorliegenden Arbeit sind ein semi-semantisches Part-of-speech Tagset, ein semi-semantisches syntaktisches Tagset, ein funktionales Tagset, Annotationsrichtlinien, eine Grammatik mit ausreichend kodierten Informationen

für das Parsen der morphologisch reichen Sprache Urdu und ein Part-of-speech annotierter Korpus, welches für das Training von Part-of-speech Taggern benutzt werden kann. Diese Ressourcen werden weiterhin erweitert und können für sprachtechnologische Prozesse, wie das Probabilistic Parsing, das Trainieren von Part-of-speech Taggern, die Disambiguierung gesprochener Sätze, Grammatikentwicklung, Sprachidentifikation, Quellen linguistischer Befragungen und psychologischer Modulierung oder Pattern Matching genutzt werden.

Acronyms

ACC Accusative

AD Anno Domini

ADJ Adjective

ADJP Adjective Phrase

ADJPQ Question Adjective Phrase

ADV Adverb

ADVP Adverb Phrase

ADVPQ Question Adverb Phrase

AH After Hijrah

C Conjunction

CAUS Causative

CBP Constraint Based Parser

CFG Context Free Grammar

CL Clause, Computational Linguistics

CM Case Marker

CMP Comparative

CARD Cardinal

CONS Concessive

COP Copula

CORD Coordinative

CORDP Coordinative Conjunction Phrase

CORR Correlative

CNF Chomsky Normal Form

CYK Cocke Younger Kasami

D Day

DAT Dative

DATEP Date Phrase

DCG Definite Clause Grammar

DEG Degree

DEM Demonstrative

DS Dependency Structure

DT Determiner

ECO Echo

EMP Emphatic

ERG Ergative

F Functional

FRAC Fractional

FUTR Future

GS Gold Standard

GUI Graphical User Interface

HDS Hyper Dependency Structure

HP Heading Phrase

HPSG Head-driven Phrase Structure Grammar

HUTB Hindi Urdu Treebank

IAA Inter Annotator Agreement

ICE-GB International Corpus of the English-Great Britain

IDE Integrated Development Environment

IMPERF Imperfective

INDF Indefinite

INF Infinitive

INST Instrumental

INTF Intensifier

JDK Java Development Kit

JJ Adjective

KON Konstanz

KP Case Phrase

KPQ Question Case Phrase

L Linked to, Lexical, Labeled

LA Labeled Attachment

LA-R Labeled Attachment with Recall

LA-P Labeled Attachment with Precision

LHS Left Hand Side

LIGHT Light Verb With Nouns or Adjectives

LIGHTV Light Verb With Verbs

ML Machine Learning

MNR Manner

MOD Modal

MODF Modifier

MRLs Morphologically Rich Languages

MRG Morphologically Rich Grammar

MST Minimum Spanning Tree

M Marker, Month

N Noun

NEG Negative

NL Non-Lexical

NLP Natural Language Processing

NN Singular Noun

NNP Proper Noun Singular

NOM Nominative

NP Noun Phrase

NPQ Question Noun Phrase

OBJ First Object

OBJ2 Second Object

OBL Oblique

ORD Ordinal

P Phrase, Pronoun

PARP Parenthetic Phrase

PASS Passive

PB Prop Bank

PBUH Peace Be Upon Him

PCFG Probabilistic Context Free Grammar

PDCG Probabilistic Definite Clause Grammar

PERF Perfective

PERS Personal

PLINK Predicate Link

POS Part Of Speech

POSS Possessive

POSTP Postposition

PP Pre/Post-position Phrase

PREP Preposition

PRES Present

PROG Progressive

PROP Proper

PS Phrase Structure

PT Particle

Q Quantifier

QP Quantifier Phrase

QW Question Word

QWP Question Word Phrase

REP Repetitive

REF Reflexive

REG Regard

REL Relative

RESULT Resultant

RHS Right Hand Side

S Sentence

SALSA The Saarbrücken Lexical Semantics Acquisition

SBAR Subordinate Clause

SBARQ Question Subordinate Clause

SBORD Subordinating

SH Solar Hijri

SPT Spatial

SQ Question Sentence

SSP Semi Semantic POS

SSS Semi Semantic Syntactic

SUB Subject

SUBTV Subjunctive

SYM Symbol

TB Treebank

TENS Tense

TMP Temporal

TTL Title

U Unit

UA Unlabeled Attachment

UDT Urdu Dependency Treebank

UP Unit Phrase

ULP Urdu Language Processing

V Verb

VALAP Vala Phrase

VAUX Verb Auxiliary

VBD Verb Past Tense

VBZ Verb 3rd Person Singular

VCMAIN Sentence Main Verb Phrase

VCP Complex Verb Phrase

VIP Infinitive Verb Phrase

VP Verb Phrase

Y Year

***** Empty Categories/Subcategories

Contents

List of Figures **xx**

List of Tables **xxii**

List of Algorithms **xxiv**

1 Introduction **1**

2 Annotation Guidelines **5**

2.1 Semi-Semantic POS (SSP) Annotation 5

2.1.1 Adjectives 6

2.1.2 Adverbs 8

2.1.3 Conjunctions 9

2.1.4 Case Markers 11

2.1.5 Date 11

2.1.6 Hadees 11

2.1.7 Interjections 12

2.1.8 Markers 12

2.1.9 Nouns 12

2.1.10 Pronouns 13

2.1.11 Postpositions 14

2.1.12 Pray 15

2.1.13 Prepositions 15

2.1.14 Particles 16

2.1.15 Quantifiers 17

2.1.16 Questions Words 17

2.1.17 Symbols 18
2.1.18 Titles 18
2.1.19 Units 18
2.1.20 Verbs 18
2.1.20.1 Copula Verbs 19
2.1.20.2 Imperfective Verbs 20
2.1.20.3 Infinitve Verbs 20
2.1.20.4 Light Verbs I 20
2.1.20.5 Light Verbs II 21
2.1.20.6 Modal Verbs 22
2.1.20.7 Perfective Verbs 23
2.1.20.8 Root Verbs 23
2.1.20.9 Subjunctive Verbs 23
2.1.20.10 Verb With Tense 24
2.1.21 Special VALA 24
2.1.22 Verb Auxiliaries 24
2.1.22.1 Imperfective Auxiliaries 25
2.1.22.2 Infinitive Auxiliaries 25
2.1.22.3 Modal Auxiliaries 25
2.1.22.4 Passive Auxiliaries 26
2.1.22.5 Perfective Auxiliaries 27
2.1.22.6 Progressive Auxiliaries 27
2.1.22.7 Root Auxiliaries 27
2.1.22.8 Subjunctive Auxiliaries 28
2.1.22.9 Tense Auxiliaries 28
2.2 Syntactic Annotation 29
2.2.1 Adjective Phrase 30
2.2.2 Question Adjective Phrase 30
2.2.3 Adverb Phrase 31
2.2.4 Question Adverb Phrase 31
2.2.5 Clause 31
2.2.6 Coordination Conjunction Phrase 32
2.2.7 Date Phrase 33

2.2.8 Parenthetic Phrase 33
2.2.9 Case Phrase 34
2.2.10 Question Case Phrase 35
2.2.11 Noun Phrase 36
2.2.12 Question Noun Phrase 37
2.2.13 Preposition/Postposition Phrase 38
2.2.14 Quantifier Phrase 39
2.2.15 Question Word Phrase 40
2.2.16 Sentence 40
2.2.17 Subordinate Clause 40
2.2.18 Question Subordinate Clause 41
2.2.19 Question Sentence 42
2.2.20 Heading Phrase 43
2.2.21 Unit Phrase 43
2.2.22 Vala Phrase 44
2.2.23 Main Verb Phrase Of The Sentence 44
2.2.24 Complex Verb Phrase 45
2.2.25 Infinitive Verb Phrase 46
2.2.26 Verb Phrase 46
2.3 Functional Annotation 47
2.3.1 1st Division of Functional Tags 48
2.3.1.1 Comparative Semantics 49
2.3.1.2 Spatial Semantics 49
2.3.1.3 Having A Semantics Of Manner 50
2.3.1.4 Temporal Semantics 51
2.3.1.5 Instrumental Semantics 52
2.3.1.6 Possessive Semantics 52
2.3.2 2nd Division of Functional Tags 53
2.3.2.1 Antecedents And Their Anaphors 53
2.3.2.2 Empty Arguments Or Categories Or Subcategories . . . 54
2.3.2.3 Relative And Correlative Clauses 56
2.3.2.4 Resultant/Consequent Clauses 58
2.3.2.5 Subject 60

2.3.2.6 First Object 60
2.3.2.7 Indirect Object 60
2.3.2.8 Oblique 62
2.3.2.9 Predicate Link 62
2.3.2.10 Modifier Or Adjunct 62
2.4 Summary 63

3 The URDU.KON-TB Treebank 64
3.1 Background 64
3.2 Related Work 68
3.3 The URDU.KON-TB Treebank 72
3.3.1 Corpus Construction 73
3.3.2 Annotation Scheme 73
3.3.2.1 Semi-Semantic POS (SSP) Tagset 74
3.3.2.2 Semi-semantic Syntactic Tagset 82
3.3.2.3 Functional Tagset 88
3.4 Comparison with the NU-FAST treebank 98
3.5 Summary 100

4 Treebank Annotation Evaluation 101
4.1 Introduction 101
4.1.1 Overview & Choices in Methods 102
4.2 Quality Control Methods for Corpus Annotations 103
4.2.1 Previous Work 103
4.2.2 Measuring Inter-Annotator Agreement (IAA) 105
4.2.2.1 Multi-Pi 105
4.2.2.2 Multi-Kappa 108
4.2.2.3 Krippendorff's Alpha 111
4.2.3 Summary 114
4.3 Evaluation 115
4.3.1 Setup 115
4.3.2 Semi-semantic POS (SSP) Tagset Evaluation 115
4.3.3 Semi-semantic Syntactic (SSS) Tagset Evaluation 118
4.3.4 Functional (F) Tagset Evaluation 122

4.4 Error Analysis and Discussion . . . 125
4.4.1 SSP Analysis & Evaluation . . . 125
4.4.2 SSS Analysis & Evaluation . . . 132
4.4.3 F Analysis & Evaluation . . . 141
4.5 Revisions to the Annotation Scheme . . . 150
4.6 Summary . . . 154

5 Urdu Parser 155
5.1 Background . . . 155
5.2 Related Work . . . 159
5.3 Motivation . . . 162
5.4 Preliminary Work . . . 163
5.5 Urdu Parser . . . 167
5.6 Analysis and Evaluation . . . 172
5.6.1 Eliminating Useless Predictions . . . 172
5.6.2 Extended Scanner . . . 173
5.6.3 Back-Pointers Calculation . . . 176
5.6.4 Building Bracketed Parse Trees . . . 178
5.6.5 Empty Productions . . . 182
5.6.6 Lexical Dynamic Behavior . . . 184
5.6.7 Limitations of Completer . . . 186
5.6.8 A Run Example . . . 190
5.7 Results . . . 191
5.8 Summary . . . 194

6 Conclusion 195

References 199

List of Figures

3.1 Phrase and dependency structure example 65
3.2 Bracket form of Figure 3.1(a). 67
3.3 A sample tree for a sentence from NU-FAST Treebank 69
3.4 A PropBank example sentence 70
3.5 C-structure for a sentence 71
3.6 F-structure for a sentence 72
3.7 A SSS annotation of example 3.2 for KP subcategories 84
3.8 A SSS annotation of NP and its subcategories 85
3.9 A SSS annotation of VCP and ADVP with subcategories 87
3.10 A SSS annotation of example KP.POSS, LIGHTV and PP. 89
3.11 Functional annotation of example 3.2 93
3.12 A functional annotation of sentences 94
3.13 A functional annotation of OBJ and OBL 96
3.14 Annotation comparison of the NU-FAST and the URDU.KON-TB treebanks 99

4.1 Agreement values and strength by Landis and Koch (1977) 108
4.2 Verb hO 'be/become' after removal of the TB tag 127
4.3 SSS annotation: a CORDP example 132
4.4 SSS annotation: with or without nested NP 133
4.5 SSS annotation: a case of CL.KER clause 136
4.6 SSS annotation: a case of VP and PP clauses 139
4.7 SSS annotation: a case of VCP 140
4.8 SSS annotation: a case of modal, progressive and passive verb auxiliaries 142
4.9 Initial F annotation: a case of SPT 144

4.10 New F annotation: a case of SPT 145
4.11 Initial annotation: a case of KP and PP as an object 147
4.12 New F annotation: a case of KP and PP as an oblique (OBL) 148
4.13 New F annotation: *sE* as POSTP.CMP 149
4.14 New F annotation: a case of -MODF 149

5.1 A verb V example from the URDU.KON-TB treebank 164
5.2 A noun phrase (NP) example from the URDU.KON-TB treebank 164
5.3 Passing up lexical information to the syntactical level 166
5.4 A CFG of the example sentence in 5.2 169
5.5 A back-pointer calculation example of the Urdu parser 177
5.6 An output of the `BUILDER()` method 181
5.7 Incomplete solution of the Urdu paser 187
5.8 Charts generated by the Urdu parser for a sentence 5.7 191
5.9 A bracketed parse tree generated by the Urdu parser for a sentence 5.7 . 192

List of Tables

2.1 A detailed version of the SSP tag set for the URDU.KON-TB treebank 7
2.2 Syntactic tagset of the URDU.KON-TB treebank. 29
2.3 Functional tag set for the URDU.KON-TB treebank 48

3.1 The main POS-Tag categories for the URDU.KON-TB treebank 74
3.2 Morphological tag set to annotate subcategories of verbs and auxiliaries 75
3.3 A detailed version of the SSP tagset for the URDU.KON-TB treebank . 76
3.4 The main syntactic-tag categories for the URDU.KON-TB treebank. . . 82
3.5 A detailed version of SSS tagset for the URDU.KON-TB treebank . . . 83
3.6 The URDU.KON-TB Functional tagset. 90
3.7 Detailed hierarchy of functional annotation in the URDU.KON-TB treebank 91

4.1 Multi- πomputation 107
4.2 Multi-κomputation 109
4.3 Karl Pearson Correlation Coefficient r 110
4.4 Overall Percentage Agreement for Multi Annotators 110
4.5 Counts of Pairwise Agreement 111
4.6 Reliability Data Matrix 112
4.7 A values by units (tokens) matrix 113
4.8 Annotators SSP tags distribution and confusion 116
4.9 SSS Reliability Data Matrix 119
4.10 A values by units (phrases) matrix 119
4.11 Annotators SSS tags distribution and confusion 120
4.12 Functional Reliability Data Matrix 122

4.13 A values by units (arguments) matrix 123
4.14 Annotators F tags distribution and confusion 124

5.1 Evaluation results of the Urdu parser 192

List of Algorithms

5.1 Earley Parsing Algorithm (Jurafsky and Martin, 2009) 158
5.2 Multi-path shift-reduce parsing model (Jiang et al., 2009) 161
5.3 A CFG extraction algorithm 168
5.4 Urdu Parser 171
5.5 Predictor 174
5.6 Scanner 175
5.7 Back Pointer 179
5.8 Builder 180
5.9 Empty Productions 184
5.10 Editor 186
5.11 Completer 189

1

Introduction

Parsing is the process of dividing a sentence into its grammatical parts along with an identification of parts and their relationship to each other. The best state-of-the-art parsing systems are based on treebank grammars (Tsarfaty et al., 2013) but unfortunately, for both constituency and dependency parsing, the treebank based technique is suffering in case of morphologically rich languages (MRLs) such as Czech (Collins et al., 1999), German (Dubey and Keller, 2003), Italian (Corazza et al., 2004), French (Arun and Keller, 2005), Modern Standard Arabic (Kulick et al., 2006), Modern Hebrew (Tsarfaty and Sima'an, 2007) and many others (Tsarfaty et al., 2010). The reason is the insufficient encoding of linguistic information in the annotation of treebanks for MRLs. The depth of information encoded in an annotation correlates with the parsing performance (Tsarfaty et al., 2013). Similarly, to obtain the promising parsing results for the MRL Urdu, a linguistically rich treebank is needed.

A treebank or a parsed corpus is a text corpus of sentences annotated with a syntactic structure. Today, many natural language processing (NLP) and machine learning (ML) applications rely on treebanks. Treebanks are heavily used in corpus linguistics for investigating syntactic phenomena or in computational linguistics for training or testing parsers. The sentences in the treebank are annotated according to an annotation scheme. The annotation scheme can include the labelings to represent morphological forms, word class, syntactic structures, semantics, grammatical arguments, co-references, etc. So, the corpus annotation is simply the addition of interpretative linguistic information to a corpus (Leech, 2005). In this thesis, to fulfill the parsing requirements of Urdu, I present the URDU.KON-TB treebank (Abbas, 2012), which is

encoded with the morphology, POS, syntactical and functional information including the handling of displaced constituents, empty categories, antecedents and anaphors, etc. This development of an annotation scheme is the fundamental step to build a treebank, for which the computational linguists then devise the annotation guidelines. In my work, the annotation structure adopted for the development of the URDU.KON-TB treebank has the combination of the PS (phrase structure) and the HDS (hyper dependency structure) annotation. Annotation issues emerged during the development are corrected and updated in the annotation guidelines. The corpus for the development of the URDU.KON-TB treebank was collected from the Urdu Wikipedia[1] and the Urdu Jang newspaper.[2]

The reliability of the treebank annotation or the annotation guidelines can be measured by calculating the agreement or the homogeneity among the annotators of the treebank. The reliability evaluation is a complex task for the treebank that contains rich information, but it is an essential part to play for the production of a quality treebank, so that the annotation can be readable. The annotation evaluation resolved most of our annotation issues except few and the annotation guidelines are updated in detail. After the annotation evaluation, the updated versions of the tag sets are named as the semi-semantic POS (SSP) tag set, the semi-semantic syntactic (SSS) tag set and the functional (F) tag set, respectively. To evaluate the annotation of the URDU.KON-TB treebank, I used the α coefficient statistical measure by Krippendorff (2004).

Annotation evaluation only guarantees the reliability of an annotation scheme of a treebank. It does not guarantee that the treebank is suitable for ML or not. The suitability of a treebank for the ML can be answered through a parser evaluation on the data of that treebank. To achieve this objective, a treebank based Urdu parser is developed and presented in this thesis according to the parsing requirements of MRLs discussed earlier. This development provides the state-of-the-art parsing results in the domain of Urdu/Hindi. The Urdu parser can help the linguists analyze the Urdu sentences computationally and it can be useful in the Urdu language processing (ULP) and ML domains. By using this parser, the size of the URDU.KON-TB treebank can also be increased.

[1] http://ur.wikipedia.org/wiki/¡.saf.ha,h awal¿

[2] `http://jang.com.pk/index.html`

The dissertation proceeds as follows. Chapter 2 describes an up-to-date annotation guidelines used for building the URDU.KON-TB treebank. These guidelines were revised after the annotation evaluation of the URDU.KON-TB treebank. The guidelines are divided into three parts mainly, which includes the POS, syntactic and the functional annotation for the purpose of readability and the simplicity. All three divisions are discussed separately along with their respective examples.

Chapter 3 provides an in-depth look at the design of the URDU.KON-TB treebank. It first provides a background of the treebanks and the annotation schemes generally and then discusses the Urdu related work. After elaborating the need of a linguistically rich Urdu treebank, it presents the development of the URDU.KON-TB treebank for Urdu in three steps, which includes the collection of sentences in the form of a corpus, manufacturing of an annotation scheme and the employment of an annotation scheme on the said corpus. At the end, it compares the linguistically rich annotation of the URDU.KON-TB treebank with the annotation of an existing NU-FAST treebank (Abbas et al., 2009) for Urdu and concludes why this URDU.KON-TB treebank with rich information was needed.

Chapter 4 evaluates the reliability of the annotation guidelines presented in Chapter 2 or evaluates the annotation reliability of the URDU.KON-TB treebank discussed in Chapter 3. It first provides a brief look on the existing statistical reliability evaluation measures and then discusses their calculation procedures along with their drawbacks. After describing why it is needed, it then shows that why Krippendorff's α (Krippendorff, 2004) statistical measure becomes my choice of annotation evaluation for the URDU.KON-TB treebank. Afterwards, the setup of annotation evaluation which includes the selection of annotators, their training, the annotated data by the annotators, etc, is explained. Having annotated data by the annotators, I then calculate the α coefficient values for the SSP, SSS and F annotation of the URDU.KON-TB treebank, which provide us the reliability of the annotation or the value for the inter-annotator agreement (IAA). This exercise comes up with annotation issues, whose error analysis with discussion is presented along with the revisions to the annotation scheme and the updated guidelines presented in Chapter 2.

In Chapter 5, the development of the Urdu parser needed to parse the MRL Urdu is presented. At first, I present a background of parsing requirements in context of MRLs, then I unveil the existing related work in the domain of Urdu. This along with some

other reasons establish the need of a treebank based Urdu parser, which has a grammar with sufficiently encoded information extracted from the URDU.KON-TB treebank. I then show the algorithmic design of the Urdu parser, which is capable to handle several parsing issues regarding the MRL Urdu. I also describe the analysis and evaluation along with the state-of-the-art parsing results, which are evaluated by comparing the Urdu parser with the three different parsers in the domain of Urdu. At the end, the chapter summarize the achievements made by the development of this Urdu parser.

Finally, in Chapter 6, I present the conclusions of my doctoral work. It is the time for some linguistics and a linguistically rich treebank is the basic requirement to parse the MRLs because the best broad coverage and robust parsers to date have grammars extracted from treebanks (Manning, 2011; Tsarfaty et al., 2013). The chapter concludes the work that why a linguistically rich Urdu treebank is needed in Chapter 3, why an up to date annotation evaluation method is necessary in case of a hierarchically designed treebank in Chapter 4 and why a morphologically rich grammar (MRG) along with the Urdu parser is needed to parse Urdu in Chapter 5.

2

Annotation Guidelines

This chapter describes the up to date annotation guidelines revised after the annotation evaluation presented in Chapter 4. The guidelines are divided into part of speech (POS), syntactic and functional annotation for easiness and simplicity. All the three divisions are discussed separately along with their respective examples. To remain on track, the annotation tags are discussed according to the order depicted for the SSP (semi-semantic POS), SSS (semi-semantic syntactic) and F (functional) tag sets in Chapter 3. The discussion of the annotation guidelines is kept concise. It can be skipped if one has already an understanding of traditional POS, syntactic and functional tags/labels and can be viewed back during the reading of Chapter 3, which possibly provides the detail of all the terms, definitions and labels used there.

2.1 Semi-Semantic POS (SSP) Annotation

The term semi-semantic (partly or partially semantic) is used with the POS because some tags are encoded with semantics but not all e.g. N.SPT (a spatial noun) tag for a word *house*, ADJ.TMP (a temporal adjective) tag for a word *previous* in *previous year*, etc. There are twenty two (22) main POS tag categories, which are displayed in Table 3.1. The description of the tags is given in the respective cells of the table. These main categories are further divided into morphological and semantical subcategories according to the Tables 3.2 and 2.3, respectively. The final and detailed version of the SSP tag set is given in Table 2.1. The dot '.' is used to add the morphological or

semantical features to the main category e.g. in V.PERF, a verb V is the main POS category like nouns, adjectives, etc, which has a perfective PERF morphology.

2.1.1 Adjectives

Adjectives are used to modify a noun or pronoun (Aarts et al., 2014; Matthews, 2007; Miller et al., 1990; Stevenson, 2010). The first main category in Table 2.1 is about ADJ (Adjective), which is divided into further five sub categories of tags included DEG (Degree), ECO (Echo), MNR (Manner), SPT (Spatial) and TMP (Temporal). The relevant POS annotations are provided in examples 2.1. Example 2.1(a) is the case of main POS category ADJ of adjective. There are some words like *tar* 'more' and *tarIn* 'most', which truly act as a degree adjective and not as degree adverb but there are some words which can play the role of a degree adverb or a degree adjective e.g. *ziyAdah* 'more/much', *bohat* 'more/much', etc, (Schmidt, 2013). Example 2.1(b) is the case of degree adjective ADJ.DEG. Example 2.1(c) is the case of reduplication[1] (Abbi, 1992; Bögel et al., 2007). As reduplication has two versions. First is discussed in a footnote below, while the other is the repetition of the original word e.g. *sAtH sAtH* 'with/along-with'. These two version are named as *echo reduplication* and *full word reduplication* by Bögel et al. (2007), which are refurbished in our annotation as ECO (echo reduplication) and REP (full word reduplication/repetition) respectively. The echo words normally starts with the letters *S* or *v* or *m*. The next examples from 2.1(d) to 2.1(f) are the cases of adjectives, which have the meaning of MNR, TMP and a SPT respectively. The addition of this MNR, TMP and SPT after the POS tag ADJ represents the semantics.

(2.1) (a) اچھا لڑکا

acHA *laRkA*
good/ADJ boy/N

'Good Boy'

(b) اہم ترین شخصیت

aham *tarIn*
important/ADJ most/ADJ.DEG
Saxs2iat
personality/N

'Most important personality'

[1]In Urdu like other South Asian languages, the reduplication of a content word is frequent. Its effect is only to strengthen the proceeding word or to expand the specific idea of a proceeding word into a general form e.g. kAm *THIk-THAk* karnA 'Do the work *right*' or kOI *kapRE-vapRE* dE dO 'Give me the *clothes or something like those'*

Table 2.1: A detailed version of the SSP tag set for the URDU.KON-TB treebank

ADJ (Adjective)	.REL (Relative)	.ROOT (Root)
.DEG (Degree)	.DEM (Demons...)	.SUBTV (Subjunctive)
.ECO (Echo)	.PERS (Personal)	.PAST (Past)
.MNR (Manner)	POSTP (Postposition)	.PRES (Present)
.SPT (Spatial)	.CMP (Comparative)	.LIGHTV (Light Verb)
.TMP (Temporal)	.MNR (Manner)	.IMPERF (Imperfective)
ADV (Adverb)	.POSS (Possessive)	.INF (Infinite)
.DEG (Degree)	.REP (Repeat)	.PERF (Perfective)
.MNR (Manner)	.SPT (Spatial)	.ROOT (Root)
.NEG (Negative)	.TMP (Temporal)	.SUBTV (Subjunctive)
.SPT (Spatial)	PRAY (Pray)	.MOD (Modal)
.TMP (Temporal)	PREP (Preposition)	.IMPERF (Imperfective)
.REL (Relative)	.MNR (Manner)	.PERF (Perfective)
C (Conjunction)	.SPT (Spatial)	.SUBTV (Subjunctive)
.CAUS (Causative)	.TMP (Temporal)	.PERF (Perfective)
.CONS (Concessive)	PT (Particle)	.REP (Repeat)
.CORD (Coordinative)	.ADJ (Adjective)	.ROOT (Root)
.CORR (Co-relative)	.EMP (Emphatic)	.REP (Repeat)
.SBORD (Subordinating)	.INTF (Intensifier)	.SUBTV (Subjunctive)
.COND (Conditional)	.RESULT (Result)	.PAST (Past)
CM (Case Marker)	Q (Quantifier)	.PRES (Present)
DATE (Date)	.ADJ (Adjective)	VALA (Vala)
.D (Day)	.CARD (Cardinal)	VAUX (Verb Auxiliary)
.M (Month)	.FRAC (Fractional)	.IMPERF (Imperfective)
.Y (Year)	.ORD (Ordinal)	.INF (Infinite)
HADEES (Hadees)	QW (Question Word)	.MOD (Modal)
INT (Interjection)	.REP (Repeat)	.IMPERF (Imperfective)
M (Marker)	.TMP (Temporal)	.PERF (Perfective)
.P (Phrase)	.SPT (Spatial)	.SUBTV (Subjunctive)
.S (Sentence)	.MNR (Manner)	.PASS (Passive)
N (Noun)	SYM (Symbol)	.IMPERF (Imperfective)
.ADJ (Adjective)	TTL (Title)	.INF (Infinite)
.MNR (Manner)	.REG (Regard)	.PERF (Perfective)
.REP (Repeat)	U (Unit)	.ROOT (Root)
.PROP (Proper)	V (Verb)	.SUBTV (Subjunctive)
.SPT (Spatial)	.COP (Copula)	.PERF (Perfective)
.TMP (Temporal)	.IMPERF (Imperfective)	.PROG (Progressive)
.REP (Repeat)	.PERF (Perfective)	.ROOT (Root)
.SPT (Spatial)	.ROOT (Root)	.SUBTV (Subjunctive)
.REP (Repeat)	.SUBTV (Subjunctive)	.FUTR (Future)
.TMP (Temporal)	.PAST (Past)	.PAST (Past)
.REP (Repeat)	.PRES (Present)	.PRES (Present)
P (Pronoun)	.IMPERF (Imperfective)	
.DEM (Demonstrative)	.REP (Repeat)	
.INDF (Indefinite)	.INF (Infinite)	
.PERS (Personal)	.LIGHT (Light)	
.POSS (Possessive)	.IMPERF (Impe...)	
.REF (Reflexive)	.INF (Infinite)	
.REP (Repeat)	.PERF (Perfective)	
.REF (Reflexive)	.PROG (Progressive)	

(c) برا ورا کام

burA	*vurA*	*kAm*
ugly/ADJ	ADJ.ECO	work/N

'Ugly work'

(d) جابرانہ حکومت

jaberaanah	*hakUmat*
cruel/ADJ.MNR	government/N

'Cruel Government'

(e) گزشتہ سال

guzaStah	*sAl*
previous/ADJ.TMP	year/N

'Preveious Year'

(f) ملتانی کھسہ

mUltAnI	*kHUsah*
multani/ADJ.SPT	shoe/N

'Multani shoe'

2.1.2 Adverbs

Adverbs can modify verbs, adjectives or other adverbs. They can also modify phrases, clauses and sentences (Aarts et al., 2014; Matthews, 2007; Miller et al., 1990; Stevenson, 2010). Adverbs are mostly used as a qualifier of the verbs but they can also be used independently. They are subcategorized into six forms presented in Table 2.1. The annotations are given in example 2.2. The main category of adverbs ADV is annotated in 2.2(a), which is further divided into five subcategories DEG (degree), MNR (manner), NEG (negative), SPT (spatial) and TMP (temporal). The final TMP has another subcategory REL for relative temporal adverb. In 2.2(b), an adverb *bohat* 'very' is used before an adjective *acHI* 'good' and it is highlighting the adjective at a certain degree, hence annotated as ADV.DEG. In 2.2(c), *biltartIb* 'respectively' behaves as an adverb and advocates a manner of order as ADV.MNR. The word *nah* 'not' is a negative adverb negating the action in 2.2(d) and it is annotated with ADV.NEG relatively. A word *sAmnE* 'front/before' is a spatial adverb and annotated as ADV.SPT in 2.2(e). The case of temporal adverb is displayed in 2.2(f), where a word *ab* 'now' is annotated as ADV.TMP. This temporal adverb is divided into another hierarchy named relative-temporal adverb, which can be seen in the last example 2.2(g). A word *jab* 'when' is given a POS tag as ADV.TMP.REL as follows.

(2.2) (a) تقریبا ساری دنیا میں

taqrIban	*sArI*	*dunIyA*	*mEN*
almost/ADV	whole/Q	world/N.SPT	in/CM

'Almost in the whole world'

(b) بہت اچھی لڑکی

bohat *acHI* *laRkI*
very/ADV.DEG good/ADJ girl/N
'Very good girl'

(c) تعداد بالترتیب ۵ اور ۶ تھی

te2dAd *biltartIb* *5* *aor* *6*
quantity/N respectively/ADV.MNR 5/Q.CARD and/C.CORD 6/Q.CARD
tHI
was/V.COP.PAST
'The quantity was 5 and 6 respectively.'

(d) عمارت مکمل نہ ہو سکی

e2emArat *mukammal* *nah* *hO*
building/N complete/ADJ not/ADV.NEG be/V.LIGHT.ROOT
saki
could/V.MOD.PERF
'The building could not be completed.'

(e) تفصیلات سامنے آئیں گی

tafs2IlAt *sAmnE* *AyIN* *gI*
details/N front/ADV.SPT come/V.SUBTV will/VAUX.FUTR
'The details will come out.'

(f) اب دیکھنا یہ ہے

ab *dEkHnA* *yE* *hE*
now/ADV.TMP to-see/V.INF this/P.PERS be/V.COP.PRES
'Now, this is to be seen.'

(g) جب یہاں کھیت ہوتے تھے

jab *yahAN* *kHEt* *hotE*
when/ADV.TMP.REL here/ADV.SPT crop-field/N.SPT be/V.IMPERF
tHE
was/VAUX.PAST
'When, there were crop fields here.'

2.1.3 Conjunctions

Conjunctions are used to connect words, phrases, clauses or sentences (Aarts et al., 2014; Matthews, 2007; Miller et al., 1990; Stevenson, 2010). The main category of conjunction C is divided into five subcategories e.g., CAUS (causative), CONS (concessive),

CORD (coordinative), CORR (correlative) and SBORD (subordinating). The last subcategory has another division of COND to represent conditional subordinate conjunction. The annotation of all divisions is presented in 2.3. Words like *cUnkEh* 'since, because', *cUnAcEh* 'so, therefore', *kIUnkEh* 'because' are candidates for a causative conjunction in a clause. An example of causative conjunction is depicted in 2.3(a). The POS annotation examples of CONS and CORD are given in 2.3(b) and 2.3(c), respectively. The word *agarcEh* 'although' is acting as a concessive conjunction in the beginning of sentence in 2.3(b), while the other word *aor* 'and' is a coordinating conjunction in 2.3(c). The word *nah* 'not,neither' as a correlative conjunction is presented in 2.3(d), in which it is annotated with C.CORR tag. The subordinating conjunction C.SBORD is annotated in 2.3(e) for a word *kEh* 'that'. The C.SBORD is divided into another subcategory proposed as COND for conditional subordinating conjunction. Its annotation for a word *agar* 'if' is presented in 2.3(f).

(2.3) (a) *SAyad* *voh* *akElA* *tHA* *kIUnkEh*
perhaps/ADV he/P.PERS alone/ADJ be/V.COP.PAST because/C.CAUS
kHAnA *hOtEl* *sE* *kHAtA* *tHA*
meal/N hotel/N.SPT from,in/CM eat/V.IMPERF be/VAUX.PAST
'Perhaps, he was alone because he used to eat his meals in a hotel.'

(b) *agarcEh* *AdmI* *kam* *tHE* *magar*
although/C.CONS men/N less/ADJ were/V.COP.PAST but/C.CORD
voh *pHir* *bHI* *jIt* *gayE*
they/P.PERS then/ADV too/PT.INTF won/V.ROOT V.LIGHTV.PERF
'Although the men were less but they had won either.'

(c) *te2dAd* *biltartIb* *5* *aor* *6*
quantity/N respectively/ADV.MNR 5/Q.CARD and/C.CORD 6/Q.CARD
tHI
was/V.COP.PAST
'The quantity was 5 and 6 respectively.'

(d) *nah* *tO* *tUm* *kHElE* *nah*
neither/C.CORR PT.EMP you/P.PERS played/V.PERF nor/C.CORR
hI *kHElnE* *diyA*
PT.INTF play/V.INF gave/V.LIGHTV.PERF
'Neither you played yourself nor you allowed to play others.'

(e) *nabI* *nE* *farmAyA* *kEh* *a2lI* *a2ilm*
prophet/N CM said/V.PERF that/C.SBORD Ali/N.PROP knowledge/N
kA *darvAzah* *hEN*
of/CM door/N.SPT is/V.COP.PRES
'The prophet stated that Ali is the door to knowledge.'

(f) *agar* *yEh* *mErA* *mAl* *hOtA*
if/C.SBORD.COND it/P.PERS my/P.POSS property/N be/V.IMPERF
tO *mEN* *xarc* *kartA*
then/PT.RESULT I/P.PERS spend/V.ROOT do/V.LIGHTV.IMPERF
'If it would be my property then I will spend it.'

2.1.4 Case Markers

Case markers distinguish the grammatical functions of words, phrases, clauses, or sentences (Aarts et al., 2014; Matthews, 2007; Miller et al., 1990; Stevenson, 2010). Urdu case markers are syntactic clitics (Butt and Sadler, 2003) and divided into different forms by Butt and King (2004) e.g., ergative, accusative, dative, possessive, etc. All Urdu case markers are annotated with a simple CM tag at POS level. Four annotated examples can be seen in 2.3(a), 2.3(e) and 2.2(a) for instrumental case marker *sE* 'from', ergative case marker *nE*, possessive case marker *kA/kI/kE* 'of' and spatial case marker *mEN/par/tak* 'in/on/at'. The different forms of case markers play an important role in identification of argument structure like subject, object, etc. The effect of different forms and their related argument structure will be discussed in Sections 2.2 and 2.3, respectively.

2.1.5 Date

The DATE tag is used to represent dates of a month e.g. 14, 2, 31, etc. This tag is divided into three subcategories, which includes DATE.D, DATE.M and DATE.Y. Annotated examples can be seen in 2.4. The days of a week, month name and a year number are represented by DATE.D, DATE.M and DATE.Y respectively.

(2.4) *aetvAr* *16* *mayI* *2004* *kO*
sunday/DATE.D 16/DATE May/DATE.M 2004/DATE.Y on/CM
'On Sunday, 16 May 2004'

2.1.6 Hadees

The Hadees is a report of deeds and saying of the prophet Muhammad (PBUH). These are tagged as HADEES in the URDU.KON-TB treebank. The Ahadees (plural of Hadees) in Arabic script in Urdu text are tagged only with this tag HADEES. The translated form of Ahadees in Urdu is annotated in a normal way. An example is depicted in 2.5 as follows. The Hadees with double quotes in the following sentence is in Arabic and hence tagged as HADEES.

(2.5)
rasUl	*nE*	*kahA*	*“*	
prophet/N	CM	said/V.PERF	M.P	
h2UsyEno-minnI-va-anA-min-al-h2UsyEn			*”*	
HADEES				M.P

‘The prophet said, “Hussain is from me and I am from Hussain”.’

2.1.7 Interjections

Interjections are the words or phrases used to exclaim, protest or command in a sentence. These are annotated with a tag INT. The example can be seen in 2.6 as follows.

(2.6)
oE	*kHAnA*	*kHAO*
OE/INT	food/N	eat/V.SUBTV

‘OE! eat the food.’

2.1.8 Markers

The markers are used to identify the boundary of phrases, clauses, or sentences as marked by punctuation. The markers are divided into two subcategories e.g. phrase markers (M.P) and sentence markers (M.S). The punctuation within the sentence like single quotes, double quotes, colon, comma, etc, are annotated with M.P, however the boundary of the sentence like full stop and question mark is annotated with M.S. The annotated example can be seen in 2.7 as follows. The comma and period is marked by M.P and M.S respectively.

(2.7)
in	*mEN*	*bHakar*	*,*		
these/P.PERS	in/CM	Bhakkar/N.PROP.SPT	comma/M.P		
layIah		*aOr*	*lOdHrAN*		*SAmil*
Layyah/N.PROP.SPT		and/C.CORD	Lodhran/N.PROP.SPT		include/N
hEN	*.*				
be/V.LIGHT.PRES	full-stop/M.S				

‘Bhakkar, Layyah and Lodhran are included in these.’

2.1.9 Nouns

The main noun tag N is divided into six subcategories, which includes adjectival noun (N.ADJ), noun having a manner (N.MNR), proper noun (N.PROP), repeated noun (N.REP)[1], spatial noun (N.SPT) and temporal noun (N.TMP). The words *cHotE* ‘younger’ and *baRE* ‘elder’ are representing people having some property of young age and old age in 2.8(a), hence both are annotated with N.ADJ. In 2.8(b), the word *t2arah2* ‘way, like, type’ is first annotated with N.MNR but when the same word is

[1]It lies in the category of full word reduplication as discussed in Section 2.1.1

repeated next then it gives the meaning of 'different types' and its repetition is annotated simply with N.MNR.REP. In 2.8(c), a subcategory N.PROP is annotated for a person name *marIyam* 'Maryam'. This subcategory is divided into two subcategories spatial and temporal, which are annotated as N.PROP.SPT and N.PROP.TMP for *panjAb* 'Punjab' and *a2Id-ul-fit2r* 'Eid festival', respectively. A common noun N is annotated in 2.8(b) for a word *taklIfEN* 'hardships'. There are some special common nouns, which can be repeated e.g. *kOrI kOrI* 'single penny'. When some noun is usually repeated then N.REP tag is used. So, this .REP along with the respective POS tag can be used to represent the presence of a repeated word. The annotation of N.SPT and N.TMP can be seen in 2.8(c) for *iz3lAa2* 'districts' and *din* 'day'. In both the subcategories, the repetition is possible for which the addition of REP with dot '.' can be used accordingly.

(2.8) (a) *cHOtE* *baRE* *sab* *xUS*
younger/N.ADJ elder/N.ADJ all/Q.ADJ happy/ADJ
hOtE *hEN*
become/V.COP.IMPERF be/VAUX.PRES
'Younger and elder all become happy.'

(b) *UnhEN* *t2arah2* *t2arah2* *kI* *taklIfEN*
they/P.PERS type/N.MNR type/N.MNR.REP of/CM hardships/N
dI *jAnE* *lagIN*
give/V.PERF go/VAUX.PASS.INF start/VAUX.SUBTV
'They were given hardships of different types.'

(c) *marIyam* *panjAb* *kE* *ba2z* *iz3lAa2*
Maryam/N.PROP Punjab/N.PROP.SPT of/CM some/Q districts/N.SPT
mEN *a2Id-ul-fit2r* *kE* *din* *gayI*
into/CM Eid-ul-Fitr/N.PROP.TMP of/CM day/N.TMP went/V.PERF
'Maryam went into some districts of Punjab on the day of Eid-ul-Fitr.'

2.1.10 Pronouns

The main category of pronoun P is divided into six subcategories P.DEM (demonstrative pronoun), P.INDF (indefinite pronoun), P.PERS (personal pronoun), P.POSS (possessive pronoun), P.REF (reflexive pronoun) and P.REL (relative pronoun). The first two subcategories P.DEM and P.INDF are annotated in 2.9(a) for words *yeh* 'this' and *kOI* 'any' respectively. The difference between P.PERS and P.DEM is this that when P.PERS refers to some person, place or thing, then this P.PERS behaves as a P.DEM like in 2.9(a). The 3rd and 4th category P.PERS and P.POSS are annotated in 2.9(b) for words *mEN* 'I' and *tumhArA* 'your' respectively. P.POSS is further divided into P.POSS.REF, which is annotated for a word *apnA* 'own' in the same sentence.

The repeated subcategory can be annotated after addition of .REP at the end. The fifth and sixth subcategory P.REF and P.REL are annotated in 2.9(c) for words *Apas* 'themselves' and *jO* 'which' respectively. The subcategory P.REL is further divided into P.REL.DEM and P.REL.PERS. These are annotated in 2.9(d) for words *jO kUcH* 'what ever' and *jIs* 'who' respectively.

(2.9) (a) *yeh* *meh2kama* *kOI* *kAm* *nahI*
this/P.DEM department/N any/P.INDF work/N not/ADV.NEG
kartA
do/V.IMPERF
'This department does not do any work.'

(b) *mEN* *tumhArA* *apnA* *bHAI* *hUN*
I/P.PERS your/P.POSS own/P.POSS.REF brother/N be/V.COP.SUBTV
'I am your own brother.'

(c) *jO* *Apas* *mEN* *moh2abat* *kI* *mIs2Al*
which/P.REL themselves/P.REF among/CM love/N of/CM example/N
hE
be/V.COP.PRES
'Which is an example of love among themselves.'

(d) *jIs* *kO* *jO* *kUcH* *milE*
who/P.REL.PERS CM what/P.REL.DEM ever/P.INDF find/V.PERF
UTHA *lEnA* *cAhIE*
pick/V.PERF take/V.LIGHTV.INF should/VAUX.MOD.PERF
'Who finds what ever, should pick it up.'

2.1.11 Postpositions

The postpositions are placed after a word to which it is grammatically related e.g. *sAtH* 'with' is a POSTP (postposition) in a postpositional phrase *Us kE sAtH* 'with him'. The postposition are divided into six subcategories hierarchically as displayed in Table 2.1. These include POSTP.CMP (comparative postposition), POSTP.MNR (postposition having a manner)[1], POSTP.POSS (possessive postposition), POSTP.REP (repetitive postposition), POST.SPT (spatial postposition) and POSTP.TMP (temporal postposition). The first two subcategories are annotated in 2.10(a) for postpositions *sE* 'than' and *t2arah2* 'like' respectively. In 2.10(b), the third and fourth subcategories are annotated for postpositions *pAs* 'have/has' and *sAtH* 'with' respectively. The last two subcategories are annotated in 2.10(c) for postpositions *qarIb* 'near' and *ba2d* 'after' respectively.

[1]The prepositions are divided into basic manner, manner by comparison and manner with a reference point by Saint-Dizier (2008) but I applied only manner in general to all related prepositions and postpositions for Urdu.

(2.10) (a) *25* *sE* *z2yAdah* *laRkE* *aslam*
25/Q.CARD than/POSTP.CMP more/ADJ.DEG boys/N Aslam/N.PROP
kI *t2arah2* *hEN*
of/CM like/POSTP.MNR be/V.COP.PRES
'More than 25 boys are like Aslam.'

(b) *xUrAk* *kE* *sAtH* *sAtH* *mErE*
food/N of/CM with/POSTP with/POSTP.REP I/P.POSS
pAs *pEsE* *bHI* *hEN*
have/POSTP.POSS money/N also/PT.INTF be/V.COP.PRES
'I have also the money along with the food.'

(c) *us* *kE* *qarIb* *h2amlE* *kE*
him/P.PERS of/CM near/POSTP.SPT attack/N of/CM
ba2d *bam* *pHatA*
after/POSTP.TMP bomb/N exploded/V.PERF
'The bomb exploded near him after the attack.'

2.1.12 Pray

The PRAY tag is used to annotate all types of prayers normally used in religious literature after the name of prophets, caliphs, and the righteous religious personalities e.g. the *alEh salAm* 'peace be upon him' is annotated with PRAY after the name of Jesus in 2.11(a) alongwith another example as follows.

(2.11) (a) *h2az3rat* *a2IsA* *alEh-salAm* *allah* *kE*
h2az3rat/TTL.REG Jesus/N.PROP AS/PRAY Allah/N.PROP of/CM
Ek *a2z4Im* *nabI* *hEN*
a/Q.CARD great/ADJ prophet/N be/V.COP.PRES
'Jesus (peach be upon him) is a great prophet of God.'

(b) *h2az3rat* *mUhammad* *s3al-lalAhO-a2laehE-va-AlEhI-vasalam* *nE*
TTL.REG Muhammad/N.PROP SAAWW/PRAY CM
h2az3rat *a2lI* *kO* *apnA* *bHAI* *banAyA*
TTL.REG Ali/N.PROP CM his/P.POSS.REF brother/N made/V.PERF
'Muhammad (peach be upon him and his descendant) made Ali his brother.'

2.1.13 Prepositions

The prepositions are placed before a word to which it is grammatically related e.g., *bE* 'without' is a PREP (preposition) in a prepositional phrase *bE mUhAr Sutar* 'a camel without a hook). Prepositions are divided into three subcategories hierarchically as displayed in Table 2.1. These include PREP.MNR (preposition having a manner), PRET.SPT (spatial preposition) and PREP.TMP (temporal preposition). The first two subcategories are annotated in 2.12(a) for prepositions *bat2Or* 'as' and *andrUnE*

'in' respectively. The last subcategory is annotated in 2.12(b) for prepositions *dOrAnE* 'during'.

(2.12) (a) *us* *nE* *bat2Or* *DrAIvar* *andrUnE* *Sehar*
he/P.PERS CM as/PREP.MNR driver/N in/PREP.SPT city/N.SPT
nOkrI *kI*
job/N do/V.PERF
'He did the job as a driver in the city.'

(b) *voh* *yahAN* *dOrAnE* *taftIS*
he/P.PERS here/ADV.SPT during/PREP.TMP investigation/N
A *giA*
come/V.ROOT go/V.LIGHTV.PERF
'He came here during the investigation.'

2.1.14 Particles

The particles can appear after a word. These are divided into four subcategories, which include PT.ADJ (adjectival particles), PT.EMP (emphatic particles), PT.INTF (Intensifying particles) and PT.RESULT (resultant particles). All the subcategories are non-inflected except the PT.ADJ, which appears after adjective, adverb, noun or pronoun and agrees with the qualifier. The first and third subcategories are annotated in 2.13(a) for the particles *sA* 'like' and *bHI* 'too'. The annotation of PT.EMP is displayed in 2.13(b) for a word *tO*. The contrastive meaning is understood by default due to usage of PT.EMP in this sentence. In 2.13(c), the annotation of PT.RESULT is given for a word *tO* 'then'.

(2.13) (a) *voh* *Ek* *nAxUSgavAr* *sA* *bandah*
he/P.PERS a/Q.CARD unpleasant/ADJ like/PT.ADJ man/N
bHI *hE*
too/PT.INTF be/V.COP.PRES
'He is like an unpleasant man too.'

(b) *ab* *maslah* *falastIn* *tO* *h2al*
now/ADV.TMP problem/N Palestine/N.PROP.SPT PT.EMP resolve/N
hO *gA*
be/V.LIGHT.ROOT will/VAUX.FUTR
'Now, the problem of Palestine will resolve (contrast: "the other problems will not" due to 'tO' effect).'

(c) *bAriS* *AyI* *tO* *mElah* *nahI*
rain/N come/V.PERF then/PT.RESULT festival/N not/ADV.NEG
hO *gA*
be/V.ROOT will/VAUX.FUTR
'If the rain comes, then the festival will not hold.'

2.1.15 Quantifiers

The quantifiers Q are used to show the amount of something. These are divided into four subcategories, which include Q.ADJ (adjectival quantifier), Q.CARD (cardinal quantifier), Q.FRAC (fractional quantifier) and Q.ORD (ordinal quantifier). In 2.14(a), the quantifiers *tamAm* 'all/whole', *har* 'every' and *dUsrA* 'second/other' are annotated with Q, Q.ADJ and Q.ORD, respectively. The remaining subcategories of quantifiers Q.CARD and Q.FRAC are annotated in 2.14(b) for words *Ek* 'one' and *cOtHAI* 'one 4th', respectively.

(2.14) (a) *tamAm* *mumAlik* *mEN* *har* *dUsrA* *Saxs*
all/Q countries/N.SPT in/CM every/Q.ADJ second/Q.ORD person/N
xUS *hE*
happy/ADJ be/V.COP.PRES
'In all countries, every second person is happy.'

(b) *mujHE* *Ek* *cOtHAI* *raqam* *dO*
me/P.PERS one/Q.CARD fourth/Q.FRAC amount/N give/V.SUBTV
'Give me one fourth amount.'

2.1.16 Questions Words

The question words QW identify a question in a sentence. These are divided into four subcategories, which include QW.REP (repeated question words), QW.TMP (temporal question words), QW.SPT (spatial question words) and QW.MNR (question words having a manner). The main category QW is depicted in 2.15(a) for a question word *kiyA* 'what'. If any question word is repeated then QW.REP can be used for annotation. The remaining three subcategories QW.TMP, QW.SPT and QW.MNR are annotated in a single sentence 2.15(b) for related question words *kab* 'when', *kidHar* 'where' and *kEsE* 'how', respectively.

(2.15) (a) *tumhArA* *nAm* *kiyA* *hE* *?*
your/P.POSS name/N what/QW be/V.COP.PRES ?/M.S
'What is your name?'

(b) *kab* *,* *kidHar* *aOr* *kEsE*
when/QW.TMP ,/M.P where/QW.SPT and/C.CORD how/QW.MNR
jAO *gE* *?*
go/V.SUBTV will/VAUX.FUTR ?/M.S
'When, where and how will you go ?'

2.1.17 Symbols

The symbols SYM include brackets, parentheses, percent symbols, currency symbols, etc. All are dealt within a single category SYM as can be seen as follows.

(2.16)

Gazvah	*fatah2*	*[*	*fatah2e*	*makkah*
Gazvah/N.PROP	fatah/N.PROP	[/SYM	fatahe/N	Makkah/N.PROP.SPT
]	*:*	*yeh*	*ramz3An*	*mEN*	*hUI*
]/SYM	:/M.P	it/P.PERS	Ramadan/DATE.M	in/CM	happen/V.PERF

'The Battle Conquest [conquest of Makkah] : It happened in Ramadan (a month name in Islamic Calendar).'

2.1.18 Titles

The titles are used to show respect or regard to personalities before addressing their names. At present it has only one subcategory TTL.REG (regard titles). Its annotation can be seen as follows.

(2.17)

h2az3rat	*ImAm*	*h2Ussyn*	*a2lEh-salAm*
his-highness/TTL.REG	religious-head/N	Hussain/N.PROP	AS/PRAY
tIsrE	*ImAm*	*hEN*
third/Q.ORD	religious-head/N	be/V.COP.PRES

'His highness, the religious head, Hussain (PBUH) is the third religious head.'

2.1.19 Units

The Unit U is used to represent different measuring units e.g., meter, liter, bar, grams, etc. The example of an annotation is given for the following sentence.

(2.18)

qEsar	*nE*	*70*	*miliyan*	*tan*	*tEl*
Qaiser/N.PROP	CM	70/Q.CARD	million/Q.CARD	ton/U	oil/N
darAmad	*kiyA*
import/N	do/V.LIGHT.PERF

'Qaiser imported 70 million ton oil.'

2.1.20 Verbs

The main verb V is divided into 11 subcategories, which are further divided into hierarchical subcategories discussed below. The hierarchical division of special word the VALA[1] and verb auxiliaries will be discussed in respective Sections 2.1.21 and 2.1.22.

[1]The VALA as a head word gives the phrase with adjectival property most likely and follows infinitive verb, nominals, adjectives, etc.

2.1.20.1 Copula Verbs

The copula verb V.COP is used to connect the subject with the subject complement or the predicate link of a sentence (Aarts et al., 2014; Butt, 1995; Matthews, 2007; Miller et al., 1990; Stevenson, 2010). For example a sentence like *The weather is horrible* contains a subject *The weather* and a predicate link as an adjective *horrible*. The predicate of the sentence is the copula verb *is*. The copula verb connects the subject with the predicate link in this sentence. The V.COP (copula verb) is divided into sex subcategories hierarchically as V.COP.IMPERF (a copula verb with imperfective morphology), V.COP.PERF (a copula verb with perfective morphology), V.COP.ROOT (a copula verb with root form), V.COP.SUBTV (a copula verb with subjunctive morphology), V.COP.PAST (copula verb with past tense) and V.COP.PRES (copula verb with present tense). The future form of copula verb itself is not possible in Urdu. In future construction, the copula verb 'be/become' always proceeds the future tense auxiliary *gA/gI/gE/gEN* 'shall/will' as can be seen in 2.19(d). The V.COP.IMPERF is annotated in 2.19(a) for a copula verb *hOtE* 'be/become' in imperfective form. Its perfective form is given in 2.19(b). The root form of another copula verb *ban* 'become' (Abbas and Raza, 2014; Raza, 2011) is presented in 2.19(c). The subjunctive form of copula verb *rahEN* 'remain' (Abbas and Raza, 2014; Raza, 2011) is annotated in 2.19(d). The copula verb with present and the past tense *hEN/tHE* 'are/were' is annotated in 2.19(e).

(2.19) (a) *SehrI* *parESAn* *nahI* *hOtE* *hEN*
citizens/N worry/ADJ not/ADV.NEG be/become/V.COP.IMPERF be/VAUX.PRES
'The citizens do not worry.'

(b) *xAhiS* *pUrI* *nah* *hUI*
desire/N fulfill/ADJ not/ADV.NEG be/become/V.COP.PERF
'The desire did not become fulfilled.'

(c) *sIrAj* *acHA* *ban* *giyA*
Siraj/N.PROP good/ADJ become/V.COP.ROOT go/V.LIGHTV.PERF
'Siraj became good.'

(d) *Ith2AdI* *kAmyAb* *rahEN* *gE*
allies/N successful/ADJ remain/V.COP.SUBTV will/VAUX.FUTR
'The allies will remain successful.'

(e) *aEsE* *lOg* *mOjOd* *hEN/tHE*
such/ADJ people/N present/ADJ be/become/V.COP.PRES/.PAST
'Such people are/were present.'

2.1.20.2 Imperfective Verbs

The imperfective verb tag V.IMPERF describes actions or states occuring generally or regularly (Schmidt, 2013). It can be identified by the inflected suffixes *tA, tI, tE, tEN* at the end of a verb. These suffixes co-exist after the root of a verb. At present, it has only one tag as V.IMPERF, however, if it is repeated then .REP can be used at the end. The example of V.IMPERF is given as follows.

(2.20) (a) *voh* *aks2ar* *cOrI* *kartE* *hEN*
they/P.PERS often/ADV stealing/N do/V.IMPERF be/VAUX.PRES
'They often do stealing.'

2.1.20.3 Infinitve Verbs

An infinitive verb V.INF can be identified through its inflected suffixes *nA, nI, nE, nEN* concatenated to the root form of a verb. The annotation example is given as follows.

(2.21) (a) *h2akUmat* *kO* *kAm* *karnA* *hE*
government/N CM work/N do/V.INF be/VAUX.PRES
'The government has to do work.'

2.1.20.4 Light Verbs I

A light verb V.LIGHT is a verb that contains a little semantic content of its own and it forms a predicate with some additional expression such as a noun or an adjective (Ahmed and Butt, 2011; Butt, 2003; Raza, 2011). It is subcategorized into eight different forms, which includes V.LIGHT.IMPERF (light verb with imperfective morphology), V.LIGHT.INF (light verb with infinitive morphology), V.LIGHT.PERF (light verb with perfective morphology), V.LIGHT.PROG (light verb with progressive morphology), V.LIGHT.ROOT (light verb with root form), V.LIGHT.SUBTV (light verb with subjunctive morphology), V.LIGHT.PAST (light verb with past tense) and V.LIGHT.PRES (light verb with present tense) as can be seen in 2.22(a)-(g) for light verbs *AtA* 'use to come', *AnA* 'to come', *AyA* 'came', *rAhA* 'remain', *A* 'come', *dO-RAEN* 'let to run' and *thA/hE* 'was/is' respectively. The respective subcategories can also be seen in Table 2.1. All the light verbs presented in annotated sentences shared their semantic content with a preceding noun N or adjective ADJ.

(2.22) (a) *mUjE* *jin* *naz4ar* *AtA* *tHA*
me/P.PERS ghost/N vision/N come/V.LIGHT.IMPERF be/VAUX.PAST
'I had used to sight the ghost.'

(b) *mUjE* *jin* *naz4ar* *AnA* *tHA*
me/P.PERS ghost/N vision/N come/V.LIGHT.INF be/VAUX.PAST
'I had to sight the ghost.'

(c) *mUjE* *jin* *naz4ar* *AyA* *tHA*
me/P.PERS ghost/N vision/N came/V.LIGHT.PERF be/VAUX.PAST
'I had sighted the ghost.'

(d) *a2li* *kO* *a2lambardArI* *kA* *a2uhdah* *h2Asil* *rAhA*
Ali/N.PROP CM flag-bearer/N of/CM designation/N gain/N
remain/V.LIGHT.PROG
'Ali had the designation of flag-bearer'

(e) *lOg* *taSadud* *bardASt* *nahI* *kar* *saktE*
people/N torture/N bear/N not/ADV.NEG do/V.LIGHT.ROOT
VAUX.MOD.IMPERF
'The people can not bear the torture.'

(f) *ham* *naqSE* *par* *naz4ar* *dORAEN* *gE*
we/P.PERS map/N onCM vision/N run/V.LIGHT.SUBTV
will/VAUX.FUTR
'We will look on the map.'

(g) *SirAj* *pUrAnE* *AdmION* *mEN* *SAmil* *thA/hE*
Siraj/N.PROP old/ADJ persons/N in/CM include/N
was/is/V.LIGHT.PAST/PRES
'Siraj was/is included in old persons.'

2.1.20.5 Light Verbs II

A light verb V.LIGHTV is a verb that contains a little semantic content of its own and it forms a predicate in presence of an additional verb (Butt, 2003), hence call verb-verb complex predicate. It is subcategorized into five different forms, which includes V.LIGHTV.IMPERF, V.LIGHTV.INF, V.LIGHTV.PERF, V.LIGHTV.ROOT and V.LIGHTV.SUBTV as can be seen in 2.23(a)-(e) for light verbs *kartA* 'used to do', *dEnA* 'to give', *liyA* 'took', *hO* 'be/become' and *jAyE* 'should go' respectively. All the light verbs presented in the annotated sentences as follows shared their semantic content with a preceding verb V or a light verb V.LIGHT.

(2.23) (a) *voh* *patHar* *luRHkA* *dEtA* *tHA*
he/P.PERS stone/N roll/V.PERF give/V.LIGHTV.IMPERF
be/VAUX.PAST
'He used to roll the stone.'

(b) *mUjHE* *jarmany* *cHOR* *dEnA*
I/P.PERS Germany/N.PROP.SPT leave/V.ROOT give/V.LIGHTV.INF
cAhIyE
should/VAUX.MOD.PERF
'I should leave Germany.'

(c) *mEN* *nE* *h2aj* *kar* *liyA*
I/P.PERS CM pilgrimage/N do/V.ROOT take/V.LIGHTV.PERF
hE
be/VAUX.PRES
'I have performed the Hajj (pilgrimage).'

(d) *janral* *mUSaraf* *kO* *fOj* *muth2arik*
general/N Musharraf/N.PROP CM army/N mobilization/N
karnA *hO* *gI*
to-do/V.LIGHT.INF be/become/V.LIGHTV.ROOT will/VAUX.FUTR
'General Musharraf will have to mobilize the army.'

(e) *yeh* *saRak* *sitambar* *mEN* *mUkammal*
this/P.DEM road/N.SPT September/DATE.M in/CM complete/ADJ
hO *jAyE* *gI*
be/become/V.LIGHT.ROOT should-go/V.LIGHTV.SUBTV VAUX.FUTR
'This road will be completed in September.'

2.1.20.6 Modal Verbs

A modal verb V.MOD expresses a scale ranging from possibility to necessity (Abbas and Nabi Khan, 2009). It is subcategorized into three morphological forms, which includes V.MOD.IMPERF (modal verb with imperfective morphology), V.MOD.PERF (modal verb with perfective morphology) and V.MOD.SUBTV (modal verb with subjunctive morphology). This category of modal verbs is different from modal auxiliaries discussed in Section 2.1.22.3, in which the main verb (predicate) of the sentence is annotated with V and the modal auxiliaries are annotated with VAUX. The examples of *cAhnA* 'want to' modified from (Facchinetti et al., 2003) are presented as follows, which contain modal verb V.MOD acting as the predicate of the respective sentence and not as an auxiliary.

(2.24) (a) *voh* *kitAb* *paRHnA* *cAhtA* *hE*
he/P.PERS book/N read/V.INF want/V.MOD.IMPERF be/VAUX.PRES
'He may wants to read the book.'

(b) *tUm* *nE* *intEqAm* *lEnA* *cAhA*
you/P.PERS CM revenge/N take/V.INF want/V.MOD.PERF
tHA
be/VAUX.PAST
'You might have wanted to take the revenge.'

(c) *voh* *intEqAm* *lEnA* *cAhEN*
they/P.PERS revenge/N take/V.INF want/V.MOD.SUBTV
gE
will/VAUX.FUTR
'They will want to take a revenge.'

2.1.20.7 Perfective Verbs

A verb with perfective morphology V.PERF can be identified through its inflected suffix e.g. *A, I, E, EN* concatenated to the root form of a verb. The annotation examples can be seen in 2.25 for a verb *kahA* 'said' in (a) and *giyA* 'went' in (b). The repetition of same verb can be annotated as V.PERF.REP. More examples can be seen in 2.3(d,e), 2.8(b), 2.9(d), 2.13(c) and 2.16.

(2.25) (a) *rasUl* *nE* *kahA*
prophet/N CM said/V.PERF
'The prophet said.'

(b) *voh* *pOlIs* *kE* *pAs* *giyA*
he/P.PERS police/N CM to/POSTP went/V.PERF
'He went to the police.'

2.1.20.8 Root Verbs

A verb with root form is a verb to which suffixes can be added (Schmidt, 2013). An annotated example can be seen in 2.26 for a verb *A* 'come', whose infinitive form is *AnA* 'to come'. More examples can be seen in 2.3(b) and 2.13(c). The repetition of same verb can be annotated as V.ROOT.REP.

(2.26) *voh* *yahAN* *dOrAnE* *taftIS*
he/P.PERS here/ADV.SPT during/PREP.TMP investigation/N
A *giA*
come/V.ROOT go/V.LIGHTV.PERF
'He came here during the investigation.'

2.1.20.9 Subjunctive Verbs

A subjunctive verb is a verb used to express hypothetical actions or conditions (Dic, 2014; Schmidt, 2013). Annotated examples can be seen in 2.2(e) and 2.15(b) for the subjunctive form of the verbs *AyIN* 'come' and *jAO* 'go' respectively.

2.1.20.10 Verb With Tense

There are sentences, the structures of which look like copular constructions but the argument requirement (subject and predicate link) for their predicates can not be fulfilled. It means that either the subject or the predicate link is missing in these type of sentences. The structure of these type of sentences is more closer to existential copula construction in English. For example, the sentence *There is the God* has an existential copular construction (Raza, 2011). The translation of this sentence in Urdu is *xUdA/God hE/is* with one argument for an existential copula verb *is*. Due to incomplete arguments in these type of sentences only, the copula verb V.COP.PRES/PAST is reduced to V.PRES/PAST for present and past tense as follows.

(2.27)
mErI	*xAhiS*	*tHI/hE*
my/P.POSS	desire/N	be/V.PAST/PRES

'It is my desire.'

2.1.21 Special VALA

The VALA is a special word in Urdu, which normally appears in a noun or an adjective phrase. It can also express the action that is going to start in a special way as can be seen in 2.28(a). Another reading of the same sentence is also mentioned. A single tag VALA is used to represent all types of *vAlA* morphological forms. Example given in 2.28(b) has a nominal reading.

(2.28) (a)
mEN	*kAm*	*karnE*	*vAlA*	*hUN*
I/P.PERS	work/N	to-do/V.INF	going/has/VALA	be/V.COP.SUBTV

'I am going to do work or I am a working person.' (two different readings)

(b)
mEN	*dUdH*	*vAlA*	*hUN*
I/P.PERS	milk/N	has/VALA	be/V.COP.PRES

'I am the milkman.'

2.1.22 Verb Auxiliaries

Verb auxiliaries VAUX denote the tense, aspect, modality, voice, mood, emphasis, etc, of the sentence predicate (Aarts et al., 2014). In Urdu, verb auxiliaries are preceded by a predicate or a complex predicate in the main verb phrase of the sentence e.g., *hO/V.COP.ROOT gayI/V.LIGHTV.PERF hE/VAUX.PRES* 'has/have become' contains a tense auxiliary VAUX.PRES along with the complex predicate *hO gayI*. Verb auxiliaries are divided into 11 subcategories discussed as follows.

2.1.22.1 Imperfective Auxiliaries

The method of identification for the imperfective auxiliary VAUX.IMPERF is the same as was discussed in Section 2.1.20.2 of imperfective verbs V.IMPERF. It is a single subcategory not contains any further divisions. An annotated example for this subcategory is given as follows.

(2.29)
kEs	*invesTigESan*	*pOlIs*	*kE*	*pAs*	*calA*
case/N	investigation/N	police/N	CM	to/POSTP	walk/V.PERF
jAtA	*hE*				
go/VAUX.IMPERF	be/VAUX.PRES				

'The case (usually) goes to investigation police.'

2.1.22.2 Infinitive Auxiliaries

The identification of infinitive auxiliaries VAUX.INF is the same as was discussed in Section 2.1.20.3 of infinitive verbs V.INF. It is also a single subcategory, whose annotated example is presented for *jAnE* 'to go' as follows.

(2.30)
Ap	*a2rab*	*qabAyl*	*mEN*	*pehcAnE*
he/P.PERS	Arab/N.SPT	tribes/N	in/CM	recognize/V.INF
jAnE	*lagE*	*tHE*		
go/VAUX.INF	take/VAUX.PERF	be/VAUX.PAST		

'He had become known in Arab tribes.'

2.1.22.3 Modal Auxiliaries

A modal auxiliary VAUX.MOD expresses a range from possibility to necessity (Abbas and Nabi Khan, 2009; Bhatt et al., 2011). It is subcategorized into three morphological forms, which includes VAUX.MOD.IMPERF (modal auxiliary with imperfective morphology), VAUX.MOD.PERF (modal auxiliary with perfective morphology) and VAUX.MOD.SUBTV (modal auxiliary with subjunctive morphology). These modal auxiliaries are different from the modal verbs discussed in Section 2.1.20.6, in which the modal verbs were acting as the predicate of the sentence but here the modal auxiliaries are following the predicate of the sentence. The examples for modal auxiliaries are as follows for *saktE* 'can', *cAhIyE* 'should' and *paREN* 'has/have to'.

(2.31) (a)
voh	*baRE*	*h2Ads2E*	*kA*	*sabab*
they/P.PERS	big/ADJ.DEG	accident/N	of/CM	reason/N
ban	*saktE*			*hEN*
become/V.COP.ROOT	can/VAUX.MOD.IMPERF			be/VAUX.PRES

'They can become the reason of a big accident.'

(b) *kAm* *xatam* *kar* *dEnA*
work/N finish/N do/V.LIGHT.ROOT give/V.LIGHTV.INF
cAhIyE
should/VAUX.MOD.PERF
'The work should be finished.'

(c) *vATar* *kOnsal* *kO* *qAnUnI* *mUSgAfIyAN*
Water/N.PROP Council/N.PROP CM regulation/ADJ anomalies/N
dUr *karnI* *paREN*
far/ADJ do/V.LIGHT.INF has/have-to/VAUX.MOD.SUBTV
gIN
will/VAUX.FUTR
'The Water Council will have to remove the regulation anomalies.'

2.1.22.4 Passive Auxiliaries

In sentences with passive auxiliaries VAUX.PASS, the theme/patient becomes the grammatical subject of the main verb. It is divided into five subcategories, which includes VAUX.PASS.IMPERF (passive auxiliary with imperfective morphology), VAUX. PASS.INF (passive auxiliary with infinitive morphology), VAUX.PASS.PERF (passive auxiliary with perfective morphology), VAUX.PASS.ROOT (passive auxiliary with root form), and VAUX.PASS.SUBTV (passive auxiliary with subjunctive morphology). The given examples have a morphological annotation of passive auxiliaries *jAtA* 'use to go', *jAnA* 'to go', *giyA* 'went', *jA* 'go' and *jAyEN* 'may go' respectively. These different forms of *jA* 'go' auxiliary are considered passive only, when they are preceded by a predicate or a complex predicate with perfective morphology (Raza, 2010).

(2.32) (a) *jAnvarON* *kO* *pAnI* *pilAyA*
animals/N CM water/N make-someone-drink/V.PERF
jAtA *hE*
go/VAUX.PASS.IMPERF be/VAUX.PRES
'The animals are watered.'

(b) *kaSmIrION* *kA* *bHI* *sOcA* *jAnA*
Kashmiri/N.SPT of/CM also/PT.INTF think/V.PERF go/VAUX.PASS.INF
cAhIyE
should/VAUX.MOD.PERF
'Kashmiri's should also be considered.'

(c) *tehsIldAr* *ka* *tabAdlah* *kiyA* *giyA*
Tehsil-officer/N of/CM transfer/N do/V.PERF go/VAUX.PASS.PERF
'The Tehsil officer has been transferred.'

(d) *sUDAn* *mEN* *nasal-kUSI* *kI*
Sudan/N.PROP.SPT in/CM genocide/N do/V.PERF
jA *rahI* *hE*
go/VAUX.PASS.ROOT continue/VAUX.PROG be/VAUX.PRES
'Genocide is being commuted in Sudan.'

(e) *kaSmIr* *sE* *fOjEN* *nikAl*
Kashmir/N.PROP.SPT from/CM armies/N takeout/V.ROOT
lI *jAyEN*
take/V.LIGHTV.PERF go/VAUX.PASS.SUBTV
'The armies may be taken out from Kashmir.'

2.1.22.5 Perfective Auxiliaries

The identification of perfective auxiliary VAUX.PERF is the same as was discussed in Section 2.1.20.7 of perfective verbs. It is an independent single subcategory, whose annotation is given in the following example.

(2.33) *lOgON* *kE* *ravaIyON* *mEN* *tabdeelI* *AtI*
people/N of/CM behavior/N in/CM change/N come/V.IMPERF
gayI
go/VAUX.PERF
'The change used to come in people's behaviors.'

2.1.22.6 Progressive Auxiliaries

The progressive auxiliary VAUX.PROG can be identified easily through its morphological form after a verb or an auxiliary. Its morphological forms include *rahA, rahE, rahI, rahIN*. An annotated example can be seen in 2.32(d) for a progressive auxiliary *rahI* 'continue'.

2.1.22.7 Root Auxiliaries

The identification of an auxiliary with a root form VAUX.ROOT is the same as discussed in Section 2.1.20.8 for verbs with root morphology. An annotated example is given as follows.

(2.34) *faqIr* *baRHtE* *jA* *rahE*
beggars/N increase/V.IMPERF go/VAUX.ROOT continue/VAUX.PROG
hEN
be/VAUX.PRES
'The beggars are increasing.'

2.1.22.8 Subjunctive Auxiliaries

A subjunctive verb auxiliary VAUX.SUBTV describes an uncertain action or state contingent on something else like permission, wish, request, etc, (Schmidt, 2013). It has no further divisions. An annotated example of subjunctive auxiliary is given as follows.

(2.35)

kiyA	*mEN*	*andar*	*AyUN?*
what/QW	I/CM	in/ADV.SPT	come/VAUX.SUBTV

'May I come in?'

2.1.22.9 Tense Auxiliaries

The tenses of auxiliary VAUX is divided mainly into three tense divisions, which include VAUX.FUTR (future tense auxiliary e.g. *gA, gI, gE, gIN*, etc.), VAUX.PAST (past tense auxiliary e.g. *tHA, tHI, tHE, tHEN*, etc.) and VAUX.PRES (present tense auxiliary e.g. *hE, hEN*, etc.). The annotation of the future tense auxiliary can be seen in 2.36(a). The annotation of past tense auxiliary is presented in 2.36(b). Similarly, the annotation of last subcategory VAUX.PRES is annotated in 2.36(c).

(2.36) (a)

ham	*naqSE*	*par*	*naz4ar*	*dORAEN*
we/P.PERS	map/N	on/CM	vision/N	run/V.LIGHT.SUBTV
gE				
will/VAUX.FUTR				

'We will look on the map.'

(b)

ham	*naqSE*	*par*	*naz4ar*	*dORA*
we/P.PERS	map/N	on/CM	vision/N	run/V.LIGHT.PERF
rahE		*tHE*		
continue/VAUX.PROG		be/VAUX.PAST		

'We were looking on the map.'

(c)

ham	*naqSE*	*par*	*naz4ar*	*dORA*
we/P.PERS	map/N	on/CM	vision/N	run/V.LIGHT.PERF
rahE		*hEN*		
continue/VAUX.PROG		be/VAUX.PRES		

'We are looking on the map.'

At this point, the semi-semantic part of speech (SSP) annotation for all tags presented in Table 2.1 has been completed along with the examples. The discussion in this Section 2.1 concludes the SSP guidelines for the URDU.KON-TB treebank. The syntactic annotation is presented in the following Section 2.2.

2.2 Syntactic Annotation

In Table 3.5, a detailed version of the semi-semantic syntactic (SSS) tag set including syntactical and semantical possibilities is given. The semantic possibilities like temporal, spatial, manner, instrumental, etc, is the part of functional annotation, which we have to discuss in the coming Section 2.3 during the representation of the related annotated trees. Therefore, in this section, only the syntactical part is presented to provide a clear understanding and to avoid repetition. The semantic possibilities will be discussed in Section 2.3, which is related to the functional annotation. The syntactic tag set given in Table 2.2 is the syntactic picture of Table 3.5 excluding semantic possibilities. Some of the main categories in Table 2.2 have further hierarchical divisions concluding thirty eight (38) syntactic tags in total. The case marking information is the syntactic information according to Kamide et al. (2003) and similarly advocated in (Butt and King, 2004) because these are clitics attached to nouns or noun phrases. Therefore the morphosyntactic features of case markers like nominative, accusative, ergative, etc, are kept at syntactic level and their grammatical relations with respect to predicate e.g., subject, first object, second object, etc, are kept at functional level which will be discussed in Section 2.3. The dot '.' is used to add syntactical features.

Table 2.2: Syntactic tagset of the URDU.KON-TB treebank.

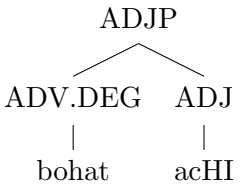
ADJP (ADJective Phrase)	.NOM (has a NOMinative case)
ADJPQ (ADJective Phrase containing Question Word)	.POSS (has a POSSessive case)
ADVP (AdVerb Phrase)	NPQ (Noun Phrase containing a Question)
ADVPQ (AdVerb Phrase containing Question Word)	.ACC (has an ACCusative case)
CL (CLause)	.NOM (has a NOMinative case)
CORDP (COoRDination Phrase)	PP (Pre/Post-Position Phrase)
.NOM (has a NOMinative case)	QP (Quantifier Phrase)
DATEP (DATE Phrase)	QWP (Question Word Phrase)
PARP (PARenthetic Phrase)	S (Sentence)
KP (case Phrase)	SBAR (subordinate clause adopted from the Penn treebank)
.ACC (has an ACCusative case)	SBARQ (subordinate clause contain Question)
.DAT (has a DATive case)	SQ (yes/no Question Sentence)
.ERG (has an ERGative case)	HP (Heading Phrase)
.POSS (has a POSSessive case)	UP (Unit Phrase)
KPQ (case Phrase containing a Question)	VALAP (special Vala Phrase)
.POSS (has a POSSessive case)	VCMAIN (Main Verb phrase of the sentence)
NP (Noun Phrase)	VCP (Verb Phrase contain Complex predicate)
.ACC (has an ACCusative case)	VIP (Verb Infinitive Phrase)
.DAT (has a DATive case)	VP (Verb Phrase at any position)

For example, in KP.ACC, the main syntactic category of case phrase KP has an accusative case marking ACC added with '.' as a syntactic feature. Each syntactic tag is discussed along with the example in the related sections as follows.

2.2.1 Adjective Phrase

The syntactic category of adjective phrase ADJP is formed, when an adjective is acting as a head word as displayed in the following annotated example 2.37.

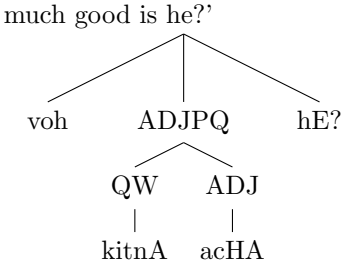
(2.37) *bohat acHI* 'very good'



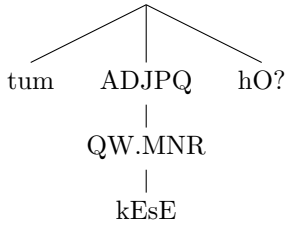
2.2.2 Question Adjective Phrase

At first, the syntactic category of a question adjective phrase ADJPQ contains a question word QW followed by an adjective ADJ like *kitnA/QW acHA/ADJ* 'How much good'. Secondly, if a question word is alone then its adjectival nature should be judged. For example, the sentence *tUm/NP kEsE/ADJPQ hO/VCMAIN?* 'How are you?' contains a noun phrase NP, a question adjective phrase ADJPQ and a main verb phrase of the sentence VCMAIN. The question word *kEsE* 'how' is making ADJPQ alone without any following adjective in contrast to the first case. The adjectival nature of *kEsE* can be judged through the knowledge or the answer replacement test. The question word *kEsE* is an adjective in Urdu (Schmidt, 2013). Moreover, the answer of the question sentence *tUm/NP kEsE/ADJPQ hO/VCMAIN?* 'How are you?' would be *mEN/NP acHA/ADJP hUN/VCMAIN* 'I am good'. An adjective *acHA* 'good' is the answer of the question word *kEsE* 'how' in this sentence. In this way, one can judge the adjectival nature of a question word. The annotated examples for both the cases are given as follows.

(2.38) (a). *voh kitnA acHA hE?* 'How much good is he?'



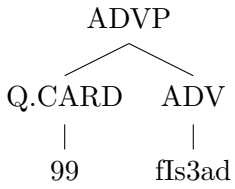
(b). *tUm kEsE hO?* 'How are you?'



2.2.3 Adverb Phrase

A syntactic category of adverb phrase ADVP is formed when an adverb is acting as a head word as displayed in the following annotated example 2.39 for *fIs3ad* 'percent'.[1]

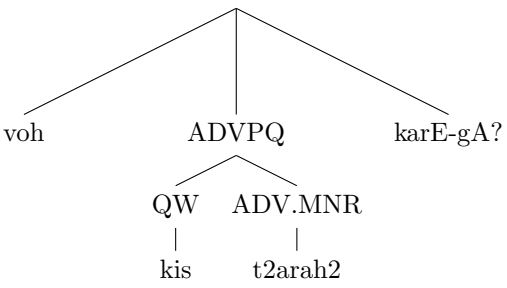
(2.39) *99 fIs3ad* '99 percent'



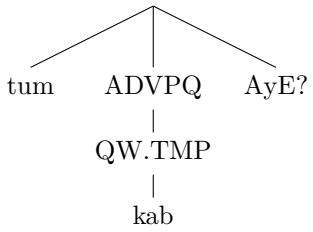
2.2.4 Question Adverb Phrase

The method of making a question adverb phrase ADVPQ is similar to ADJPQ discussed in Section 2.2.2 except the head word would be an adverb ADV and not an adjective. The annotated examples of a question word followed by an adverb *kis/QW t2arah2/ADV.MNR* 'how' and an alone question word having adverbial nature *kab* 'when' are presented as follows.

(2.40) (a). *voh kis t2arah2 karE-gA?* 'How will he do?'



(b). *tUm kab AyE?* 'When did you come?'



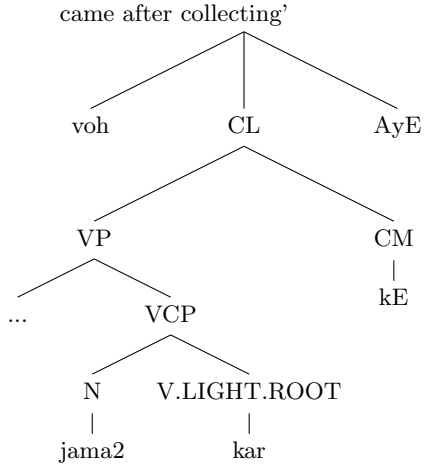
2.2.5 Clause

A clause CL acting as a modifier/adjunct in a sentence can be identified by a combination of *kar/V.ROOT kE/CM* 'after doing' or by another combination of a verb followed by *kar/V.ROOT*. The rule for building a syntactic structure for both the clauses is to break the combination e.g. for *kar*/V.ROOT *kE*/CM, a verb phrase VP will be made first by considering the POS tag of *kar/V.ROOT* and then a clause CL is introduced over it. Similarly, for other combination of verb followed by *kar* 'do' like

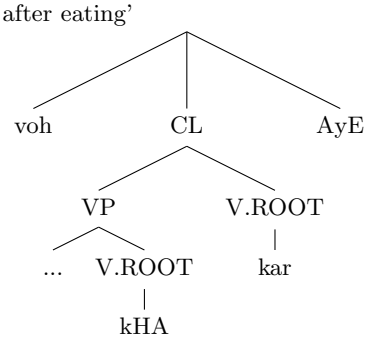
[1]The lexicon is available at `www.cle.org.pk/oud/`.

in *kHA/V.ROOT kar/V.ROOT* 'after eating', One should introduce a VP at the verb *kHA* 'eat' and then CL at *kar* 'do'. Both clauses are structured out of the VCMAIN, which is the main verb phrase of the sentence. The respective annotated examples for both the cases are given as follows. There is no difference between the given two trees except the presence of complex predicate *jama2 kar* 'do collection'. In case of complex predicate, one should introduce VCP (complex predicate phrase) first before the introduction of VP.

(2.41) (a). *voh ...jama2 kar kE AyE* 'They came after collecting'

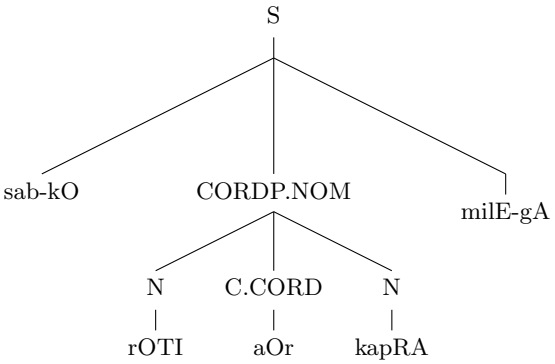


(b). *voh ...kHA kar AyE* 'They came after eating'

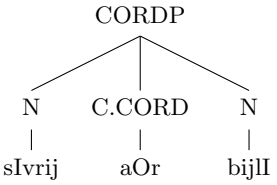


2.2.6 Coordination Conjunction Phrase

(2.42) (b). *sab kO rOTI aOr kapRA milE gA* 'All will get the food and cloth'



(a). *sIvrij aOr bijlI* 'sewerage and electricity'



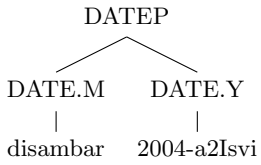
A coordination conjunction phrase CORDP is built when a coordination conjunction is found in it. This category is divided into another subcategory CORDP.NOM (coordination conjunction phrase has a nominative case) for its representation at root

S level only. In the annotation of the URDU.KON-TB treebank, the functional labeling is compulsory at root S level. So, when a coordination conjunction is used among nominals and the phrase is the candidate for argument structure e.g. subject, object, etc, then normally the arguments of the sentence in the URDU.KON-TB treebank has some case information e.g. nominative, accusative, dative, etc, discussed in the beginning of Section 2.2. To fulfill this purpose, this category of CORDP.NOM is introduced to handle the coordination conjunction phrases with nominative case at root S level, which are also acting as an argument of the sentence. The phrase in example 2.42(a) is the simple case of CORDP by assuming that it is not attached to the root S but below that level. The coordination phrase CORDP.NOM attached to the root S in example 2.42(b) is acting as the object of the sentence with the nominative case information.

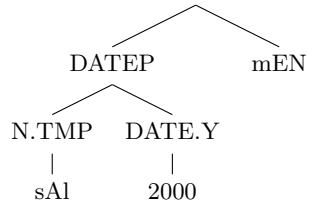
2.2.7 Date Phrase

A date phrase DATEP is formed, when a date e.g. day, month, year, etc., is found to be the head word. An annotated example is presented as follows.

(2.43) (a). *disambar 2004-a2Isvi* 'December AD 2004)



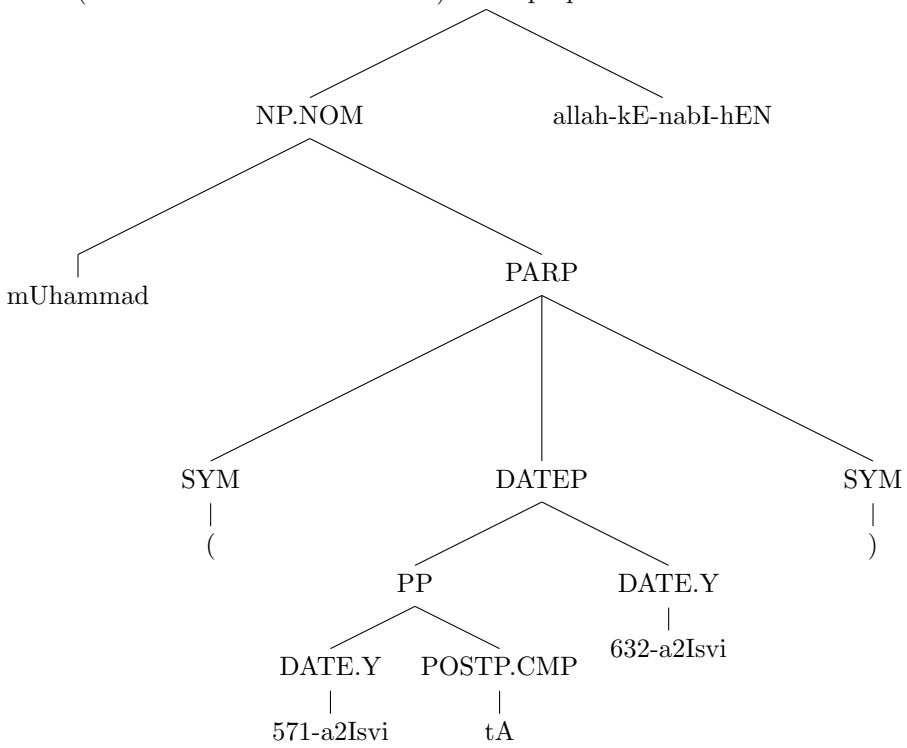
(b). *sAl 2000 mEN* 'in year 2000)



2.2.8 Parenthetic Phrase

In the sentences, some times a description or related information of an object, an event, a person, etc., is provided between brackets or parenthesis. This type of parenthetical information does not take part in the argument structure of the sentences. To deal with this type of information, a parenthetic phrase PARP is introduced, whose annotated example is given as follows. Irrelevant things at this point, which are not annotated in the example tree belongs to the arguments structure and the predicate of the sentence. The phrase NP.NOM is the subject of the sentence, while the PARP contains only the parenthetical data in the sentence. Similarly, the *allah kE nabI* 'the prophet of God' is the PLINK (predicate link) of the copula verb *hEN* 'be/become'. These all new terms and labels are discussed in the coming sections.

(2.44) *mUhammad (570-a2Isvi yA 571-a2Isvi tA 632-a2Isvi) allah kE nabI hEN*
'Muhammad (570 AD or 571 AD to 632 AD) is the prophet of God'

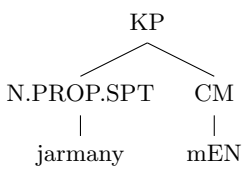


2.2.9 Case Phrase

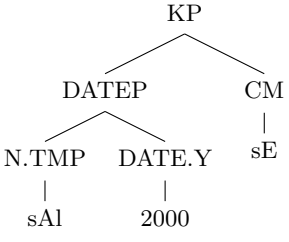
The case phrase KP is built when a case marker CM becomes the head word in a phrase. These case markers include *kA, kI, kE, kO, nE, mEN, par, tak* and *sE*. The *kA, kI, kE* 'of' are the possessive case markers. The *kO* 'to' can be an accusative or a dative, *nE* is an ergative case marker, while *mEN, par, tak* are locative case markers. The final *sE* is an instrumental case marker. A detailed study of case markers can be seen in (Butt and King, 2004). The KP is divided into four major subcategories, which include KP.ACC (case phrase has an accusative case), KP.DAT (case phrase has a dative case), KP.ERG (case phrase has an ergative case) and KP.POSS (case phrase has a possessive case). The annotation examples are given as follows.

(2.45)

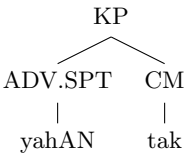
(a). *jarmany mEN* 'in Germany'



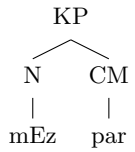
(b). *sAl 2000 sE* 'from year 2000'



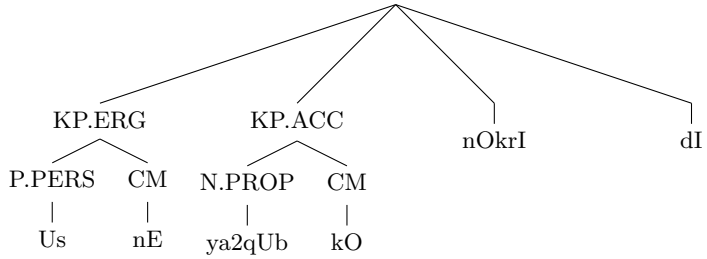
(c). *yahAn tak* 'to here'



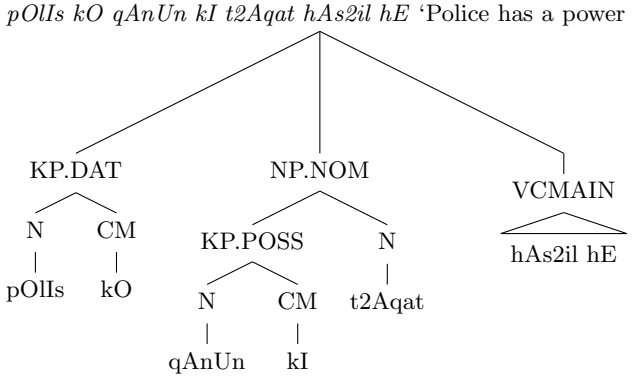
(d). *mEz par* 'on the table'



(e). *Us nE ya2qUb kO nOkrI dI* 'He gave a job to Jacob'



(f). *pOlIs kO qAnUn kI t2Aqat hAs2il hE* 'Police has a power of law'

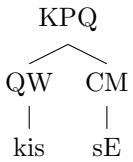


2.2.10 Question Case Phrase

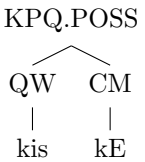
When a KP has a question word followed by a case marker, then it becomes KPQ. A case phrase having a question word KPQ has the same subcategories as presented in previous Section 2.2.9 of the case phrase. However, only the KPQ and the KPQ.POSS is displayed based on the evidence from the URDU.KON-TB treebank. The subcategory KPQ.POSS keeps a possessive case marker as a head word along with the question word. The examples of KP given in 2.45 can be repeated with KPQ by replacing the dependent words of KP with question words e.g. *kis* 'who/whom/which' as can be seen in 2.46(c,d).

(2.46)

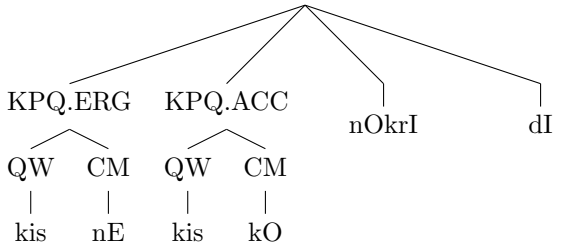
(a). *kis sE?* 'from whom?'



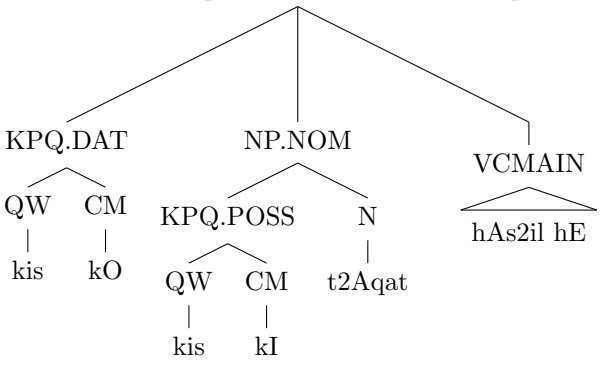
(b). *kis kE?* 'to whom?'



(c). *kis nE kis kO nOkrI dI* 'Who gave the job to whom?'



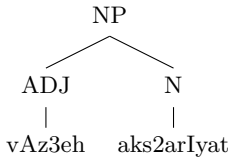
(d). *kis kO kis kI t2Aqat hAs2il hE* 'Who has a power of whom/which?'



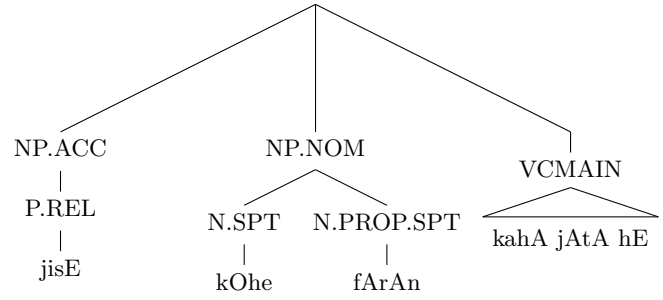
2.2.11 Noun Phrase

The noun phrase NP is built when a nominal becomes the head word in a phrase. The NP is divided into four major subcategories, which include NP.ACC (noun phrase has an accusative case), NP.DAT (noun phrase has a dative case), NP.NOM (noun phrase has a nominative case) and NP.POSS (noun phrase has a possessive case). These four subcategories along with the major syntactic category NP conclude five divisions, whose annotated examples can be seen as follows.

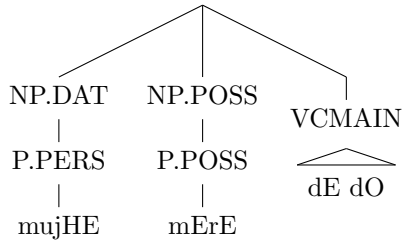
(2.47) (a). *vAz3eh aks2arIyat* 'clear majority'



(b). *jisE kOhe fArAn kahA jAtA hE* 'Which is called the mountain of Faran'



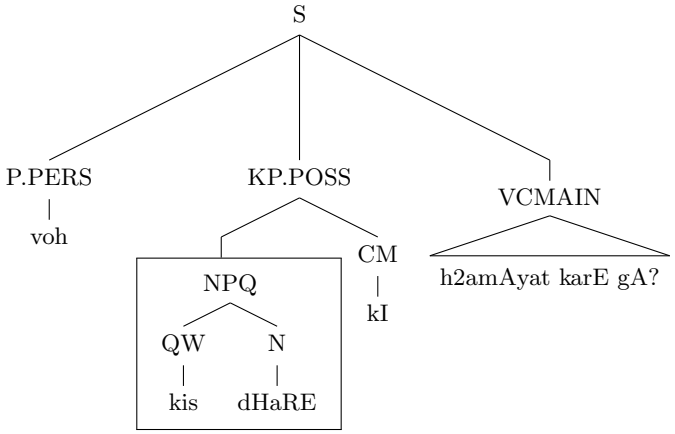
(c). *mujHE mErE dE dO* 'Give me mines'



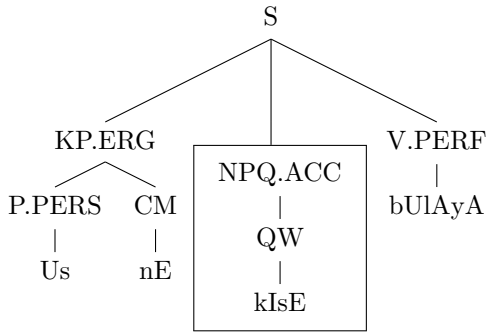
2.2.12 Question Noun Phrase

When an NP has a question word followed by a noun, then it is annotated as a NPQ. It has again the same subcategories as presented in Section 2.2.11 of noun phrase except the .POSS (possessive), which is not possible for NPQ as NPQ.POSS because the question words in Urdu may have accusative, dative and nominative inflectional forms but don't have any possessive inflectional form e.g. *kisE* 'to whom' is the inflectional form of question word *kis* 'whom/who', which can be accusative NPQ.ACC or dative NPQ.DAT as can be seen in 2.48(b,d), respectively but cannot be possessive. To use the question word *kis* in possessive form , one can use the possessive case marker *kA/kI/kE* 'of' after the *kis*, which then becomes the case of question case phrase KPQ.POSS and not the case of NPQ.POSS. The main category NPQ not attached to root S along with the subcategory NPQ.NOM (question noun phrase has a nominative case) attached to root S are presented in 2.48(a,c), respectively. The category NPQ along with its subcategories NPQ.ACC and NPQ.NOM in Table 2.2 are depicted on the basis of evidence from the URDU.KON-TB treebank. The last subcategory NPQ.DAT is possible but not exist in the URDU.KON-TB treebank due to its limited size. Topic related parts of the trees highlighted in squares are annotated completely as compared to other parts and this is the normal practice in the whole annotation guidelines.

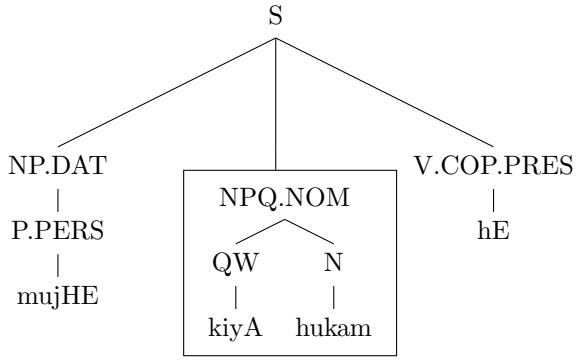
(2.48) (a). *voh kis dHaRE kI h2amAyat karE gA?* 'He will support which group?'



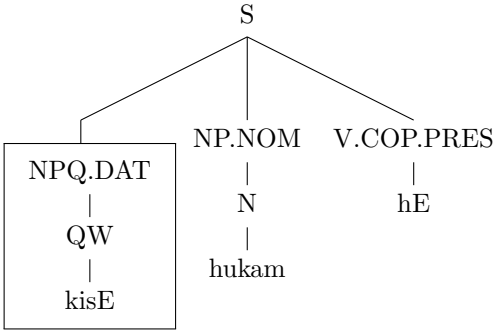
(b). *Us nE kIsE bUlAyA?* 'Whom he called?'



(c). *mujHE kiyA hukam hE?* 'What orders do I have?'



(d). *kisE hukam hE?* 'Whom has the order?'

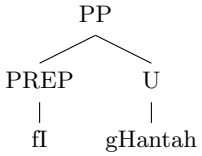


2.2.13 Preposition/Postposition Phrase

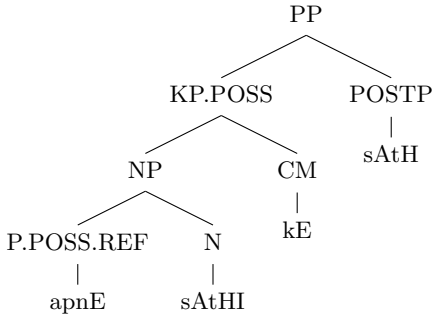
A preposition or a postposition phrase PP is formed when a preposition or a postposition is acting as a head word in that phrase. Some nouns, adverbs and case markers can behave like a postposition (Butt and King, 2004; Rizvi, 2008) in a sentence. When a noun or an adverb follows a possessive case marker *kE* with oblique form, then that noun or adverb would be a postposition as can be seen in 2.49(b,d) for a noun *sAtH* 'support' and an adverb *qarIb* 'near', respectively. The case marker *sE* 'from' can act like a postposition in comparative situations (Rizvi, 2008) like in 2.49(c). Prepo-

sition precedes the word without any case marker as given in 2.49(a) and are itself prepositions.

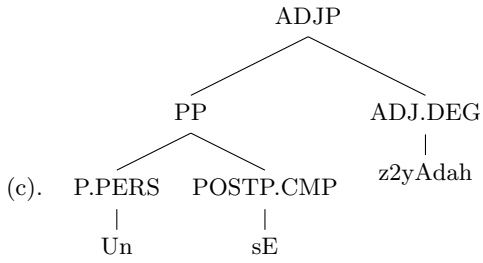
(2.49) (a). *fI gHantah* 'per hour'



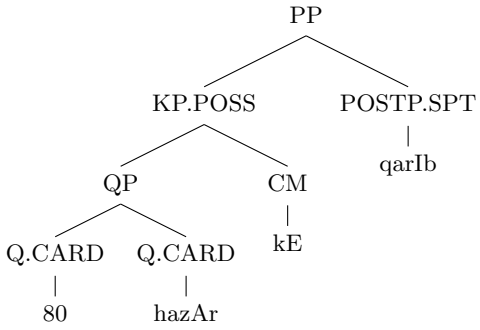
(b). *apnE sAtHI kE sAtH* 'with his friend'



(c). *Un sE z2yAdah* 'more than them'



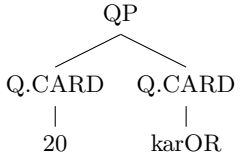
(d). *80 hazAr kE qarIb* 'about 80 thousand'



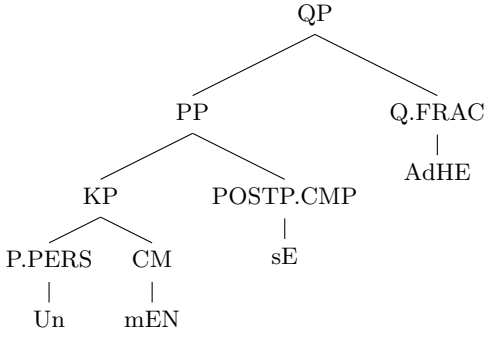
2.2.14 Quantifier Phrase

A quantifier phrase QP is formed when a subcategory of quantifier e.g. cardinal Q.CARD, ordinal Q.ORD, fractional Q.FRAC, etc. is acting as a head word in that phrase. The examples are presented as follows.

(2.50) (a). *20 karOR* '10 million'

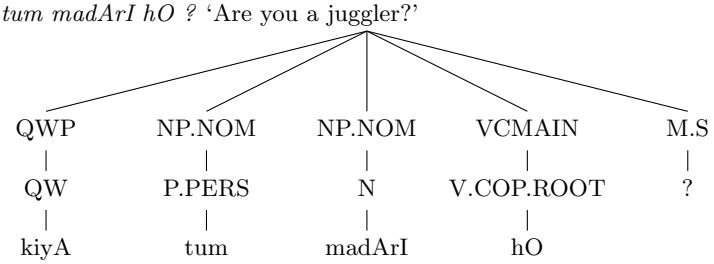


(b). *Un mEN sE AdHE* 'half of them'



2.2.15 Question Word Phrase

(2.51) *kiyA tum madArI hO ?* 'Are you a juggler?'

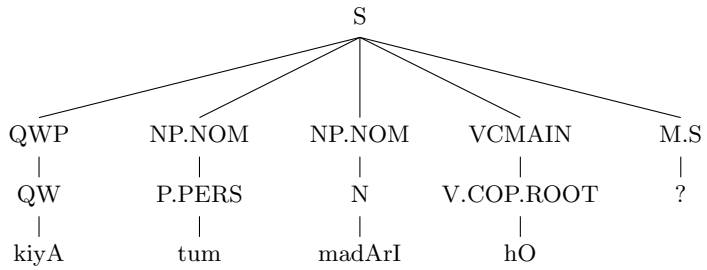


A question word phrase QWP is introduced when a question word QW becomes the head word in a yes/no question sentence e.g. *kiyA* 'what/do/does/did'. This syntactic category is unique among the categories discussed earlier in Sections 2.2.2, 2.2.4, 2.2.10, and 2.2.12 for ADJPQ, ADVPQ, KPQ and NPQ, respectively because the question word QW in this category always appears independently as can be seen in 2.51. This QWP category is normally introduced when the question sentence has an answer in the form of yes or no. The annotated example is given in 2.51.

2.2.16 Sentence

A sentence S represents the root of a tree structure. The complete annotated examples of sentences presented in earlier sections mostly do not have a root label. It was not displayed intentionally until we reach here at this point of discussion because we are going according to the tags order in Table 2.2. From now to onwards, the root of the tree structure will be displayed with a label S as can be seen in the updated version of example 2.51 as follows.

(2.52) *kiyA tum madArI hO ?* 'Are you a juggler?'

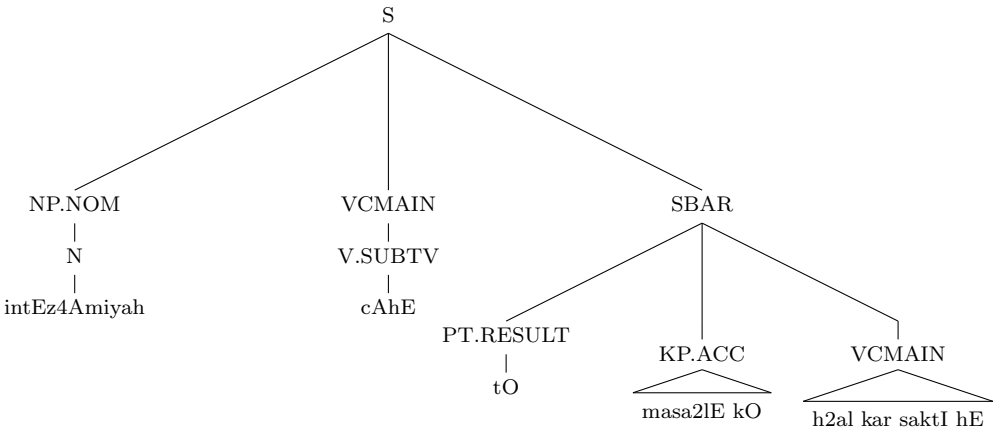


2.2.17 Subordinate Clause

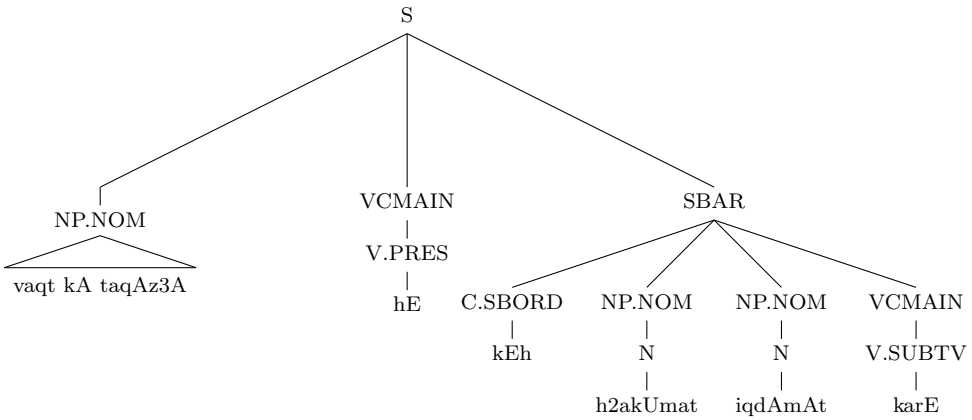
A subordinate clause is introduced when the subordinate, causative, concessive, correlative or resultant clause appears in the sentence with respective conjunctions. A

Penn treebank style of representing subordinating clause SBAR is adopted. The representations of SBAR indicated by a resultant particle PT.RESULT and a subordinate conjunction C.SBORD are given in 2.53(a,b), respectively.

(2.53) (a) *intEz4Amiyah cAhE tO masa2lE kO h2al kar saktI hE* 'If the administration wants, then they can solve the problem'



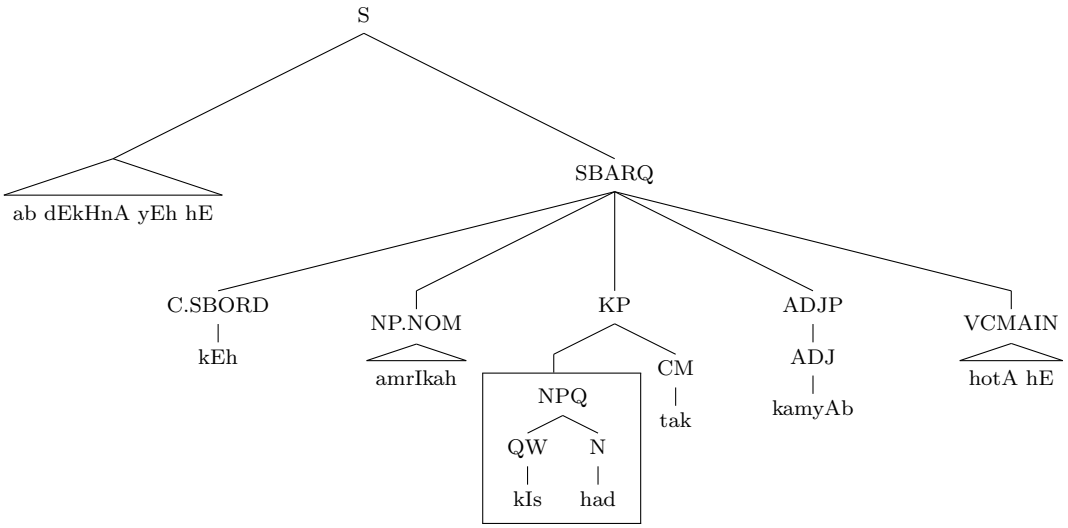
(b) *vaqt kA taqAz3A hE kEh h2akUmat iqdAmAt karE* 'It is the demand of time that the government should do actions.'



2.2.18 Question Subordinate Clause

A subordinate clause with a sense of question along with a question word or a nested question phrase is labelled with SBARQ. An annotated example is presented in 2.54.

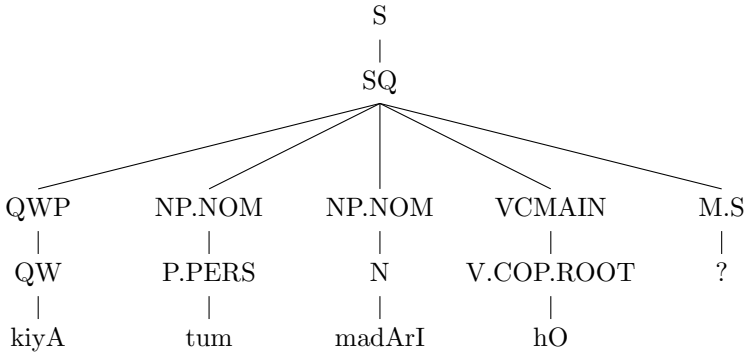
(2.54) *ab dEkHnA yEh hE kEh amrIkah kIs had tak kamyAb hotA hE* 'It is to see now whether America becomes successful at which level?'



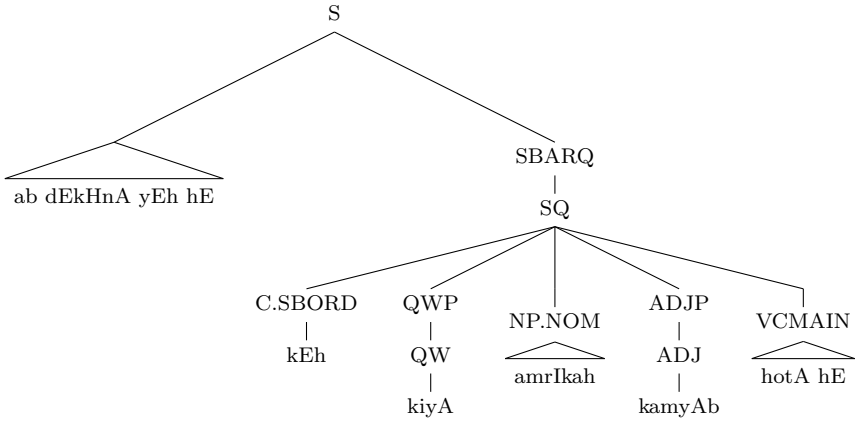
2.2.19 Question Sentence

A yes/no question sentence or clause is labelled with SQ. If a sentence has no any nested clauses then SQ is simply labelled inside the root S. If a sentence has a question subordinate clause SBARQ then the SQ will be annotated inside the SBARQ. The annotated examples for both the cases are presented in 2.55.

(2.55) (a). *kiyA tum madArI hO ?* 'Are you a juggler?'



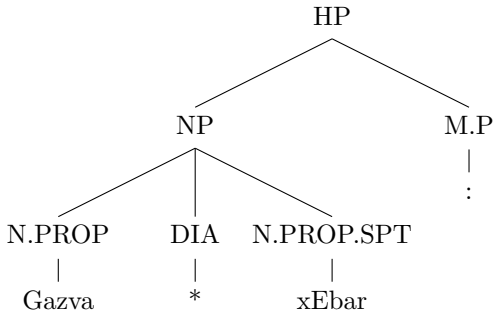
(b). *ab dEkHnA yEh hE kEh kiyA amrIkah kamyAb hotA hE* 'It remains to be seen whether America will be successful?'



2.2.20 Heading Phrase

The heading of a paragraph, article, etc, is annotated as HP (heading phrase). The example is given in 2.56.

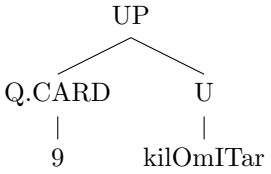
(2.56) *Gazva xEbar :* 'Battle of Khybar:'



2.2.21 Unit Phrase

When a unit like meter, liter, etc, behaves as a head word in a phrase, then that phrase is annotated as UP (unit phrase). The example can be seen in 2.57.

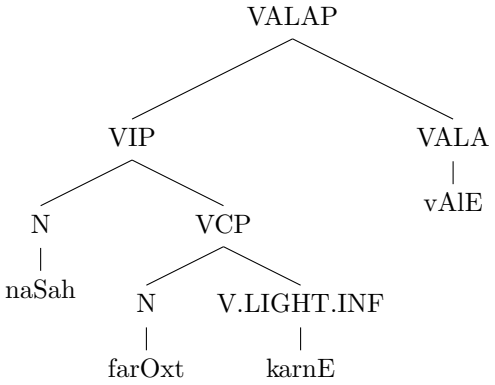
(2.57) *9 kilOmITar* '9 kilometer'



2.2.22 Vala Phrase

When a special VALA discussed in Section 2.1.21 behaves as a head word in a phrase, then that phrase is labelled as VALAP. The example can be seen in 2.58.

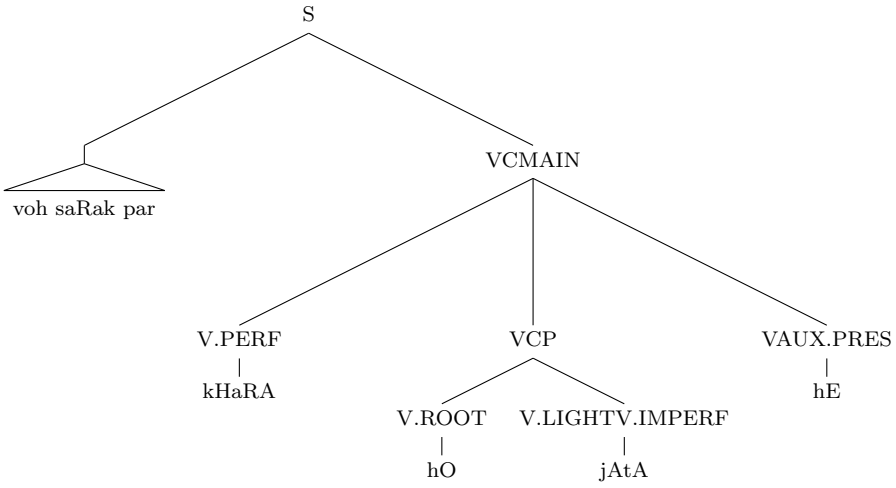
(2.58) *naSah farOxt karnE vAlE* 'drug sellers'



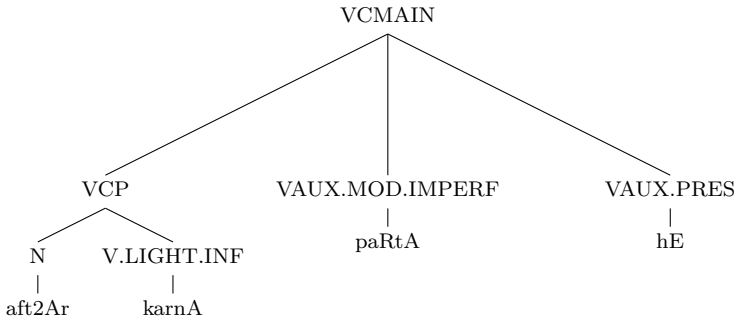
2.2.23 Main Verb Phrase Of The Sentence

The main verb phrase of the sentence which contains the predicate along with the aspectual and tense auxiliaries is annotated as VCMAIN. It can also contain complex predicates phrase VCP discussed in the forthcoming section. An annotated example can be seen in 2.59.

(2.59) (a). *voh saRak par kHaRA hO jAtA hE* 'He goes to stand on the road'



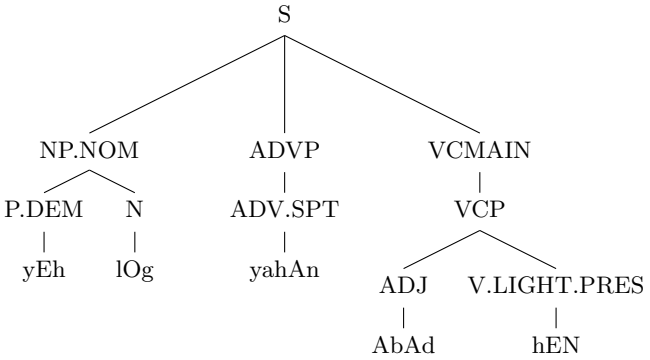
(b). Un kO rOzah saRak par *aft2Ar karnA paRtA hE* 'They have to break the fast on the road'



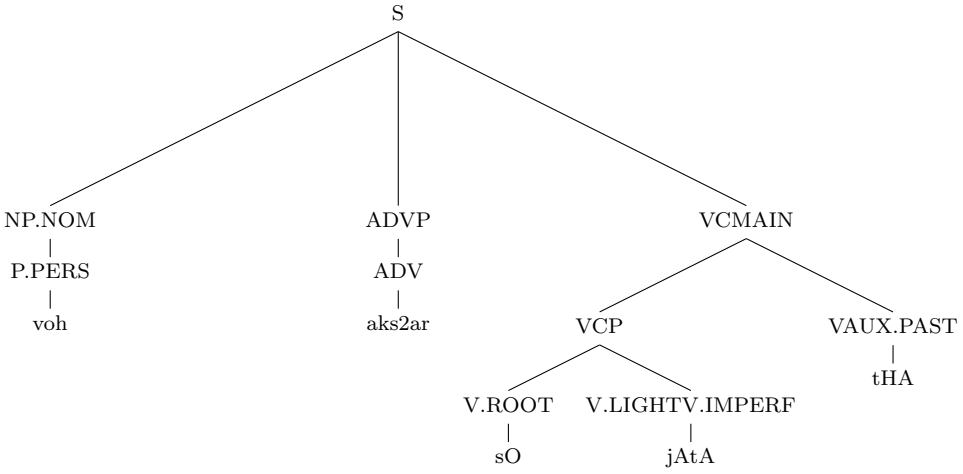
2.2.24 Complex Verb Phrase

A phrase containing a complex predicate is annotated as a VCP. In this phrase, the light verb becomes the head word. The examples of its annotation are presented in 2.60.

(2.60) (a). *yEh lOg yahAn AbAd hEN* 'These people live here'



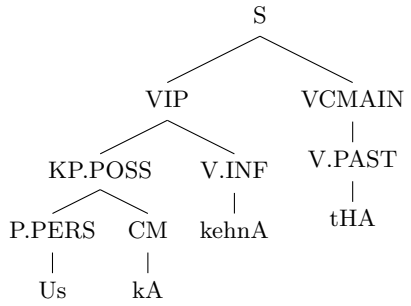
(b). *voh aks2ar sO jAtA tHA* 'He had often slept'



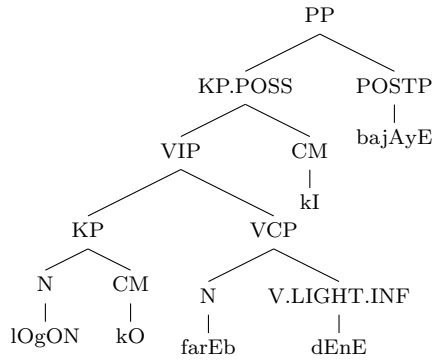
2.2.25 Infinitive Verb Phrase

When an infinitive verb becomes a head word in a phrase, then that phrase is annotated as VIP. A VIP can have a nominal sense and can behave as an argument for the predicate of a sentence like VIP acting as the subject of the sentence in 2.61(a). Other example is given in 2.61(b).

(2.61) (a). *Us kA kehnA tHA* 'It is his saying'



(b). *lOgON kO farEb dEnE kI bajAyE*
'instead of dodging the people'

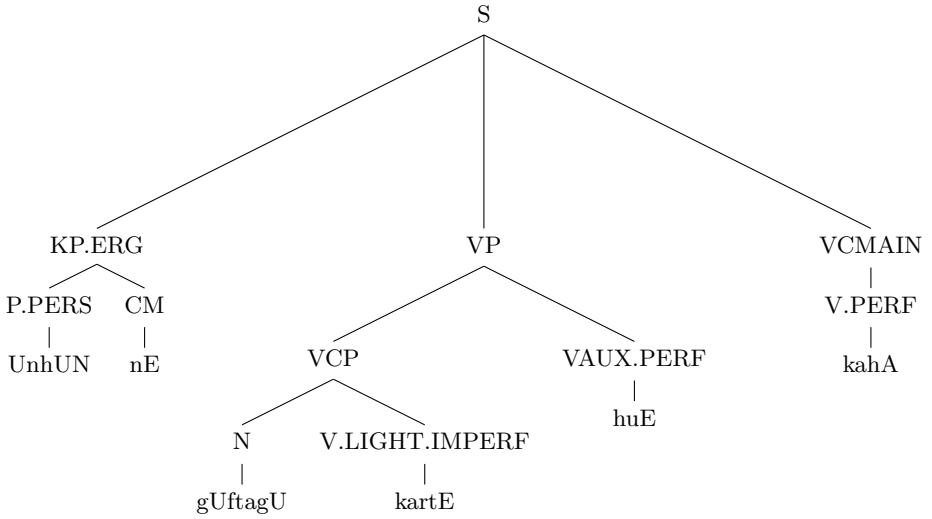


2.2.26 Verb Phrase

In the annotation of the URDU.KON-TB treebank, a linguistically motivated VCMAIN adopted from the Urdu LFG grammar project[1] acts as the main verb phrase of the sentence. It also contains the predicate of the sentence as discussed in Section 2.2.23. On the other hand, a VP (verb phrase) behaves as a modifier/adjunct in a sentence outside the VCMAIN and it also contains a verb as a head word. An example can be seen in 2.62.

[1] `http://ling.uni-konstanz.de/pages/home/pargram_urdu/`

(2.62) *UnhUN nE gUftagU kartE huE kahA* 'They said while talking'



This section concluded the syntactic annotation of the URDU.KON-TB treebank. All the subsections contained annotated examples for each syntactic category and its subcategories. The semantic features which can be added up from lexical level to syntactic level are not discussed due to simplicity. It is the thing which is closely related to functional annotation. So, it will be discussed in the forthcoming Section 2.3. The semantic features were already discussed at POS level in Section 2.1, However, these will be explored further along with the complete functional features presented in Table 2.3.

2.3 Functional Annotation

There are eighteen (18) labels in this functional tag set displayed in Table 2.3 divided into semantical, clausal, grammatical and miscellaneous types. The tags in the functional (F) tag set have two major divisions. In first division a small class of semantic tags exists, which includes CMP (comparative), SPT (spatial), MNR (manner), TMP (temporal), INST (instrumental), and POSS (possessive). The second division includes tags for clauses as CORR (correlative), REL (relative), and RESULT (resultant) and the tags for labeling of subcategorization frames (grammatical labels) like SUB (subject), OBJ (first object), OBJ2 (second object), OBL (oblique), PLINK (predicate link) and a MODF (modifier/adjunct). Similarly, the numbers in miscellaneous part are for labeling of antecedents, which are linked with L to their related anaphors. The empty

categories or subcategories are represented with an asterisk '*' symbol. The precedence order of the URDU.KON-TB treebank annotation from lexical level to functional level is the POS > Syntactic > 1st Division of F > 2nd Division of F. At POS level '.' is the only symbol used to connect POS with semantical and morphological subcategories e.g. N.SPT (the POS of noun N with spatial SPT semantics) or V.LIGHT.PERF (the POS of verb V acting as a light verb LIGHT has a perfective PERF morphology). At syntactical level, '.' symbol is used to connect syntactical subcategories and '-' symbol is used to add all functional labels including semantical tags after the syntactic tags. An example of a complete syntactic tag along with the functional labels according to the precedence order of the annotation is NP.NOM-SPT-SUB. Here, a syntactic tag NP with a morphosyntactic information of nominative case .NOM is the syntactic part, -SPT from the 1st division of the functional tags is giving us information that the phrase has a spatial sense, while -SUB from the 2nd division of functional tags is giving us information that this NP is acting as a subject of the sentence. A description of all the functional tags with examples is presented as follows. Due to the possibility of long examples, the bracketed notations rather than the tree structures are presented in this section.

Table 2.3: Functional tag set for the URDU.KON-TB treebank

Semantic labels	
CMP (Comparative)	POSS (Possessive)
INST (Instrumental)	SPT (Spatial)
MNR (Manner)	TMP (Temporal)
Clause labels	
CORR (Correlative clause of a sentence)	RESULT (Resultant clause of a sentence)
REL (Relative clause of a sentence)	
Grammatical labels	
MODF (Modifier of a sentence)	OBL (Oblique of a sentence)
OBJ (Object of a sentence)	PLINK (Predicate-Link of a sentence)
OBJ2 (Second Object of a sentence)	SUB (Subject of a sentence)
Miscellaneous labels	
* (for representing empty entities)	L (for linking of anaphors to antecedents)
1,2,3... (for labeling of antecedents)	

2.3.1 1st Division of Functional Tags

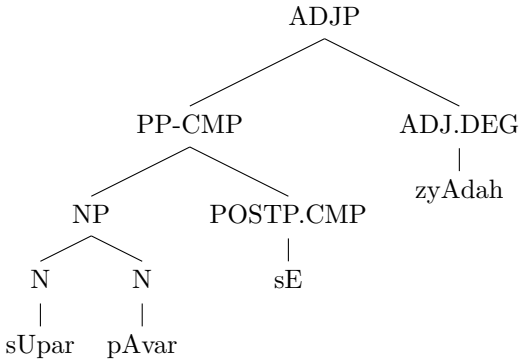
As discussed, this division contains six different semantic tags, which includes CMP, SPT, MNR, TMP, INST, and POSS. These tags also exist at the POS level with '.'

symbol. To apply these tags at syntactic level functionally, a rule has to be followed. If a phrase contains a head word with a tag from this 1st division of functional tags, then the tag will always be annotated with '-' at the phrase level. Moreover, if that semantically/functionally annotated phrase is followed by a case marker then this semantic/functional feature will be passed up to the following phrase. It means that these tags in the 1st division of functional tags will always be annotated with '-' symbol at phrase/syntactic level and not with '.' symbol. The examples are as follows.

2.3.1.1 Comparative Semantics

A phrase PP can be annotated as PP-CMP (comparative pre/post-positional phrase) in the presence of a comparative postposition POSTP.CMP acting as a head word. The annotated example is given in 2.63.

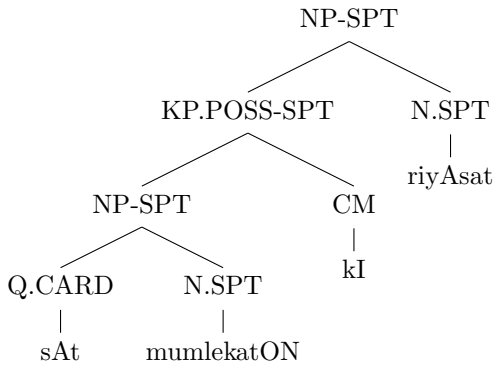
(2.63) *sUpar pAvar sE zyAdah* 'more than the super power'



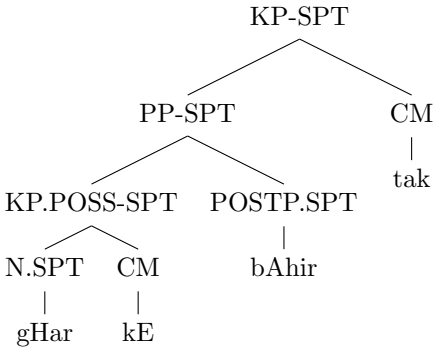
2.3.1.2 Spatial Semantics

A spatial phrase can be annotated with a -SPT semantics in presence of a spatial head word with .SPT as a part of its POS tag. The annotated examples are given in 2.64.

(2.64) (a). *sAt mumlekatON kI riyAsat* 'The state of 7 substates'



(b). *gHar kE bAhir tak* 'To the outside of the house'

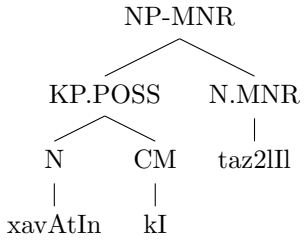


2.3.1.3 Having A Semantics Of Manner

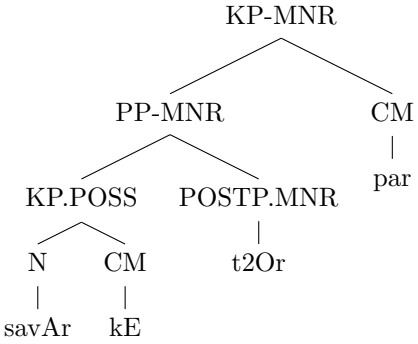
A phrase can be annotated with a -MNR semantics in presence of a head word with .MNR as a part of its POS tag. A phrase with this kind of tagging means it has a semantics of manner in it. The annotated examples are given in 2.65.

(2.65) (a). *isIt2arah2* 'similarly'

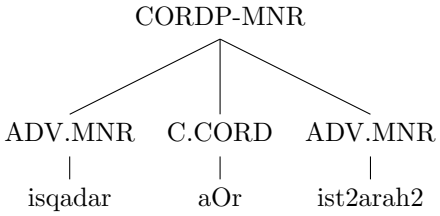
(b). *xavAtIn kI taz2lIl* 'humiliation of women'



(c). *savAr kE t2Or par* 'like a rider'



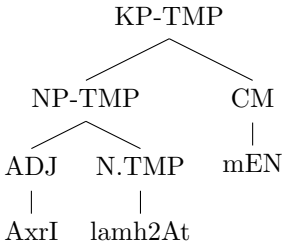
(d). *isqadar aOr ist2arah2* 'This much and in this way'



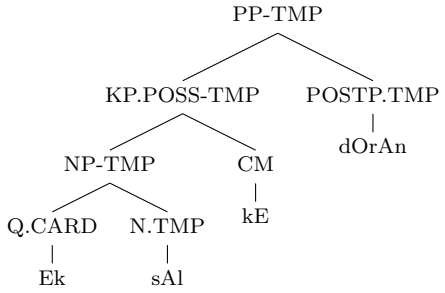
2.3.1.4 Temporal Semantics

A temporal phrase can be annotated with a -TMP semantics in presence of a temporal head word with .TMP as a part of its POS tag. The annotated examples are given in 2.66.

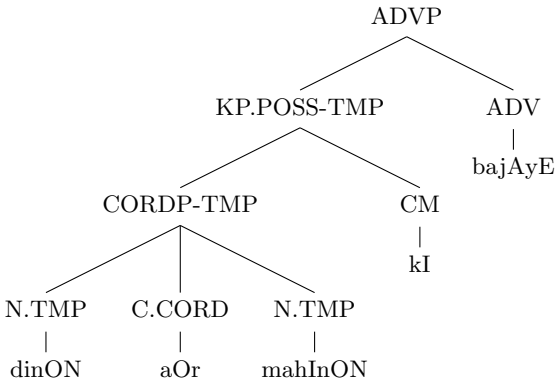
(2.66) (a). *AxrI lamh2At mEN* 'In the last moments'



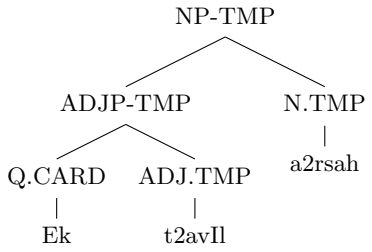
(b). *Ek sAl kE dOrAn* 'During a year'



(c). *dinON aOr mahInON kI bajAyE* 'Instead of days and months'



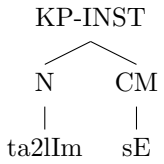
(d). *Ek t2avIl a2rsah* 'A long period'



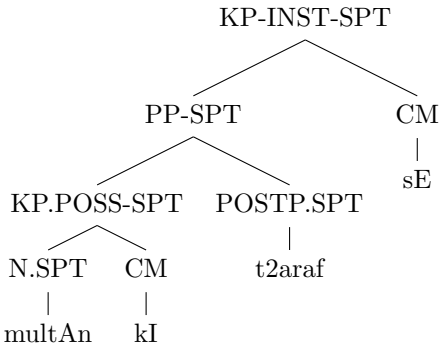
2.3.1.5 Instrumental Semantics

An instrumental phrase can be annotated with a -INST semantics in the presence of a instrumental case marker *sE* 'from' as a head word except when the case marker *sE* appears as a comparative postposition POSTP.CMP as discussed in Section 2.3.1.1. The annotated examples are given in 2.67.

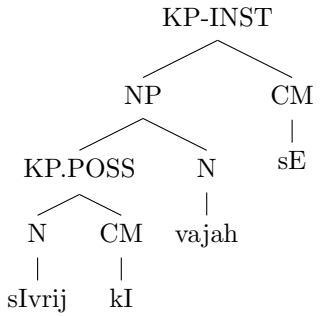
(2.67) (a). *ta2lIm sE* 'From education'



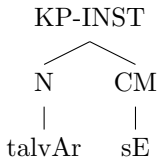
(b). *multAn kI t2araf sE* 'From the side of Multan'



(c). *sIvrij kI vajah sE* 'Due to sewerage'



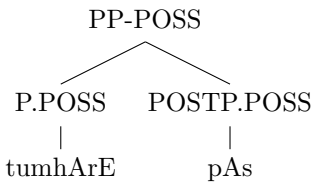
(d). *talvAr sE* 'With sword'



2.3.1.6 Possessive Semantics

A possessive phrase can be annotated with a -POSS semantics in presence of the head word with .POSS as the part of its POS tag. An annotated example of possessive postposition is given in 2.68.

(2.68) *tumhArE pAs* 'in your possession'



2.3.2 2nd Division of Functional Tags

As discussed, this division contains twelve different functional tags. The miscellaneous part in Table 2.3 includes tags for antecedent numbering annotated with 1, 2, 3, etc, empty categories & subcategories represented with an asterisk '*', linking of pro-forms or anaphors, clauses and arguments. The tags for clauses are CORR (correlative), REL (relative), and RESULT (resultant). Similarly, the tags for labeling of arguments are like SUB (subject), OBJ (first object), OBJ2 (second object), OBL (oblique), PLINK (predicate link) and a MODF (modifier/adjunct). To apply these tags at syntactic level functionally, a '-' dash symbol is used after the annotation of the 1st division of functional tags. The examples are as follows.

2.3.2.1 Antecedents And Their Anaphors

In a sentence like ***Ali*** *said* *he* *eats mangoes*, the proper noun in bold is the antecedent of the underlined pro-form or anaphor. In our annotation, the antecedents are numbered as -1,-2,-3, etc., in a sentence, which are then linked with their related anaphors or pro-forms by using a label L for linking to numbered antecedents as -L-1, -L-2, -L-3, etc., respectively. A bracketed example can be seen in 2.69. In this sentence, *voh* 'they' is an anaphor/pro-form to an antecedent *amrIkI afvAj* 'American armies'. In Urdu, a pro-form can be absent in this type of sentences. When this is the situation, then it can be annotated by an empty phrase like (NP.NOM-SUB-L-1 *) or (NP.NOM-SUB-L-1 (P.PERS *)). Here the asterisk '*' represents an empty argument or an empty lexical item, respectively.

(2.69) *amrIkI afvAj xavAtIn kI bEh2UrmatI kar rahI hEN aOr voh lUtmAr mEN mas3rUf hEN .*
'American armies are disgracing women and they are busy in looting and killing.'

```
( S
     ( NP.NOM-SUB-1
          ( ADJ.SPT amrIkI ) ( N afvAj )
     )
     ( KP.POSS-OBL
```

```
                ( N xavAtIn ) ( CM kI )
        )
        ( VCMAIN
            ( VCP
                ( N.MNR bEh2UrmatI ) ( V.LIGHT.ROOT kar )
            )
            ( VAUX.PROG rahI ) ( VAUX.PRES hEN )
        )
        ( C.CORD aOR )
        ( NP.NOM-SUB-L-1
            ( P.PERS voh )
        )
        ( KP-MNR-OBL
            ( N.MNR lUTmAr ) ( CM mEN )
        )
        ( ADJP-PLINK
            ( ADJ mas3rUf )
        )
        ( VCMAIN
            ( V.COP.PRES hEN )
        )
        ( M.S . )
)
```

2.3.2.2 Empty Arguments Or Categories Or Subcategories

An argument or a category or a subcategory or an entity in general can be absent in a sentence in Urdu. To represent such an absence, the '*' symbol is used during the annotation of a sentence as discussed in Section 2.3.2.1. For example, the most frequent phenomenon of diacritic's absence can be annotated as (DIA *) in a compound word like *qarz/N */DIA h2asnah/ADJ* 'charitable/interest-free loan' in 2.70. Similarly, a personal pronoun *voh* 'they' is the anaphor of the antecedent *bErozgAr ta2lImyAftah nOjavAnON* 'Unemployed educated youth', which is also absent in the beginning of the subordinate clause in 2.70. However, to annotate the absence of anaphor and its anaphoric relation using the numbers and the link L, it can be annotated as an empty argument (NP.NOM-SUB-L-1 (P.PERS *)) in the subordinate clause and link it with antecedent (KP.ACC-SUB-1 ...) in the matrix clause.

(2.70) *bErozgAr ta2lImyAftah nOjavAnON kO qarz3 h2asnah farAham kiyA jAE tAkEh kArObAr SUrUa2 kar sakEN .*
'Unemployed educated youth should be provided with interest free loans so that they can start a business.'

```
( S
     ( KP.ACC-SUB-1
          ( NP
               ( ADJP
                    ( ADJ bErozgAr ) ( ADJ ta2lImyAftah )
               )
               (    N nOjavAnON   )
          )
          ( CM kO )
     )
     ( NP.NOM-OBJ
          ( N qarz3E ) ( DIA * ) ( ADJ h2asnah )
     )
     ( VCMAIN
          ( VCP
               ( N farAham ) ( V.LIGHT.PERF kiyA )
          )
          ( VAUX.PASS.SUBTV jAE )
     )
     ( SBAR
          ( C.SBORD tAkEh )
          ( NP.NOM-SUB-L-1
               ( P.PERS * )
          )
          ( NP.NOM-OBJ
               ( N kArObAr )
          )
          ( VCMAIN
               ( VCP
                    ( N SUrUa2 ) ( V.LIGHT.ROOT kar )
               )
               ( VAUX.MOD.SUBTV sakEN )
          )
          ( M.S . )
     )
)
```

2.3.2.3 Relative And Correlative Clauses

In a sentence, a clause which contains a relative word e.g. *jO, jab, jEsA, jis-jagah,* etc., is known as a relative clause. On the other hand, a clause which contains a correlative version of relative words e.g. *voh, tab, vEsa, Us-jagah,* etc., is called a correlative clause (Butt et al., 2007; Schmidt, 2013). A relative and correlative clause are annotated with -REL and -CORR tags, respectively. The examples are presented in 2.71 as follows.

(2.71) (a). *h2azrat jibrAIl a2lEh-salAm nE allah kI jAnib sE jO pEhlA pEGAm insAn kO puhncAyA voh yEh hE* 'The first message by the side of God which Gabriel delivered to human being is this.'

```
( S
    ( KP.ERG-SUB
        ( NP
            ( TTL.REG h2azrat ) ( N.PROP jibrAIl ) ( PRAY a2lEh-salAm )
        )
        ( CM nE )
    )
    ( KP-INST-SPT-MODF
        ( PP-SPT
            ( KP.POSS
                ( N.PROP allah ) ( CM kI )
            )
            ( POSTP.SPT jAnib )
        )
        ( CM sE )
    )
    ( NP.NOM-OBJ
        ( P.REL jO )
        ( NP
            ( Q.ORD pEhlA ) ( N pEGAm )
        )
    )
    ( KP.DAT-OBJ2
        ( N insAn ) ( CM kO )
    )
    ( VCMAIN
        ( V.PERF puhncAyA )
    )
    ( SBAR-CORR
        ( NP.NOM-SUB
```

```
                ( P.PERS voh )
            )
            ( NP.NOM-PLINK
                ( P.PERS yEh )
            )
            ( VCMAIN
                ( V.COP.PRES hE )
            )
            ( M.S . )
        )
)
```

(b). *yEh hidAyat dI tHI kEh jang kA jO bHI fEslah hO voh hamArI qismat hO gA* 'It was advised that whatever decision would be made with respect to the war that would be our fate.'

```
( S
   ( NP.NOM-SUB
      ( P.PERS yEh )
   )
   ( NP.NOM-OBJ
      ( N hidAyat )
   )
   ( VCMAIN
      ( V.PERF dI ) ( VAUX.PAST tHI )
   )
   ( SBAR-REL
      ( C.SBORD kEh )
      ( NP.NOM-SUB
         ( KP.POSS
            ( N jang ) ( CM kA )
         )
         ( NP
            ( P.REL.DEM jO ) ( PT.INTF bHI ) ( N fEslah )
         )
      )
      ( VCMAIN
         ( V.SUBTV hO )
      )
      ( SBAR-CORR
         ( NP.NOM-SUB
            ( P.PERS voh )
```

```
            )
            ( NP.NOM-OBJ
               ( P.POSS hamArI ) ( N qismat )
            )
            ( VCMAIN
               ( V.SUBTV hO ) ( VAUX.FUTR gA )
            )
            ( M.S . )
         )
      )
   )
```

2.3.2.4 Resultant/Consequent Clauses

A resultant or consequent clause -RESULT is dependent on a matrix clause. A resultant particle like *tO* 'then', compound adverbs like *isliyE* 'therefore' and *pHir bHI* 'nevertheless', a causal conjunction like *cUnAcEH* 'so', etc, are the candidates to introduce this kind of resultant clause. The examples of the annotation can be seen in 2.72.

(2.72) (a). *voh pIcHE haT gayE tHE IsliyE mArE gayE* 'They had stepped back, therefore they had been killed'.

```
( S
   ( NP.NOM-SUB-1
      ( P.PERS voh )
   )
   ( ADVP-SPT-MODF
      ( ADV.SPT pIcHE )
   )
   ( VCMAIN
      ( VCP
         ( V.ROOT haT ) ( V.LIGHTV.PERF gayE )
      )
      ( VAUX.PAST tHE )
   )
   ( SBAR-RESULT
      ( ADVP-MNR-MODF
         ( ADV.MNR IsliyE )
      )
      ( NP.NOM-SUB-L-1 * )
      ( VCMAIN
```

```
            ( V.PERF mArE ) ( V.LIGHTV.PERF gayE )
         )
         ( M.S . )
      )
   )
```

(b). *bHutO vazIrE Azam ban gayE tO mulk sE Gurbat kA xAtmah hO jAyE gA* 'If Bhutto became the prime minister, then poverty would be abolished from the country.' (In Urdu, *agar/C.SBORD.COND* 'if' is optional, therefore it is absent in the beginning of the conditional matrix clause, which is annotated with '*')

```
( S
   ( C.SBORD.COND * )
   ( NP.NOM-SUB
      ( N.PROP bHutO )
   )
   ( NP.NOM-PLINK
      ( N vazIrE) ( ADJ Azam )
   )
   ( VCMAIN
      ( VCP
         ( V.COP.ROOT ban ) ( V.LIGHTV.PERF gayE )
      )
   )
   ( SBAR-RESULT
      ( PT.RESULT tO )
      ( KP-INST-SPT-MODF
         ( N.SPT mulk )( CM sE )
      )
      ( NP.NOM-SUB
         ( KP.POSS
            ( N Gurbat )( CM kA )
         )
         ( N xAtmah )
      )
      ( VCMAIN
         ( VCP
            ( V.ROOT hO ) ( V.LIGHTV.PERF jAyE )
         )
         ( VAUX.FUTR gA )
      )
   )
```

)

2.3.2.5 Subject

A study about the subject of a sentence in Urdu can be seen in (Butt et al., 1999; Mohanan, 1994). A subject of a sentence is represented with a -SUB functional tag in the URDU.KON-TB treebank. The annotation having a subject can be seen in all examples from 2.69 to 2.72 respectively. The phrases which can become a subject according to the evidence from the URDU.KON-TB treebank includes coordination conjunction phrase, case phrase, noun phrase including question noun phrase, preposition/postposition phrase, quantifier phrase, vala phrase and verb infinitive phrase. The case of the subject can be nominative, accusative, dative, ergative, and possessive (Butt et al., 1999; Mohanan, 1994; Tiwari, 2004). The possible combination of subject -SUB with 1st division of functional tags can be seen in Table 3.7.

2.3.2.6 First Object

First object of a sentence is represented with an -OBJ functional tag in the URDU.KON-TB treebank. The annotation of the sentences having a first object can be seen in 2.70 and 2.71. The phrases which can become a first object according to the evidence from the URDU.KON-TB treebank include question adverb phrase, coordination conjunction phrase, case phrase, noun phrase including question noun phrase, and verb infinitive phrase. The case of a first object can be nominative and accusative (Butt et al., 1999; Mohanan, 1994). The possible combination of object -OBJ with 1st division of functional tags e.g. spatial, temporal, manner, etc., can be seen in Table 3.7.

2.3.2.7 Indirect Object

A second object of a sentence is represented with an -OBJ2 functional tag in the URDU.KON-TB treebank. The annotation of the sentences having a second object can be seen in 2.71(a). The phrases which can behave as a second object according to the evidence from the URDU.KON-TB treebank includes case phrases and noun phrases. The case of a second object can be a nominative (in special cases like in 2.73) or a dative (Abbas, 2012; Butt et al., 1999; Mohanan, 1994). The possible combinations of a second object with 1st division of functional tags e.g. spatial, temporal, manner, etc., can be seen in Table 3.7.

(2.73) (a). *mua2SrE kA har fard axlAqI pAbandIUN sE mubarrA AzAdI kA h2asUl apnA bunyAdI haq gardAntA hE* 'Every man of society considers his fundamental right to have an independence without the ethical constraints'.

```
( S
     ( NP.NOM-SUB
          ( KP.POSS
               ( N mua2SrE ) ( CM kA )
          )
          ( NP
               ( Q.ADJ har ) ( N fard )
          )
     )
     ( NP.NOM-OBJ2
          ( PP
               ( KP-INST
                    ( NP
                         ( ADJ.MNR axlAqI ) ( N pAbandIUN )
                    )
                    ( CM sE )
               )
               ( POSTP mubarrA )
          )
          ( NP
               ( KP.POSS
                    ( N AzAdI ) ( CM kA )
               )
               ( N h2asUl )
          )
     )
     ( NP.NOM-OBJ
          ( P.POSS.REF apnA )
          ( NP
               ( ADJ bunyAdI ) ( N haq )
          )
     )
     ( VCMAIN
          ( V.IMPERF gardAntA ) ( VAUX.PRES hE )
     )
     ( M.S . )
)
```

2.3.2.8 Oblique

An argument which is not acting as the subject, object or the predicate link of a sentence predicate but providing some obligatory means, due to which it can not be treated as an adjunct/modifier (Butt et al., 1999). The example in 2.69 is the representation of an oblique argument of a sentence. This oblique is annotated with -OBL in the URDU.KON-TB treebank. The phrases which can become an oblique according to evidence from the URDU.KON-TB treebank include adjective phrase, question adjective phrase, adverb phrase, case phrase, question case phrase, preposition/postposition phrase and verb infinitive phrase. The case of an oblique can be a possessive and the possible combinations of an oblique with 1st division of functional tags include spatial, temporal, manner and instrumental.

2.3.2.9 Predicate Link

In the presence of a copula verb, two arguments are essential which include a subject and a predicate link. For example a sentence like *The weather is horrible* contains a subject *The weather* and a predicate link (PLINK) as an adjective *horrible*. The predicate of the sentence is the copula verb *is*. The copula verb builds a relation between a subject and a predicate link. The predicate link is annotated with -PLINK tag in the URDU.KON-TB treebank annotation. The examples can be seen in 2.69, 2.71(a) and 2.72(b). The phrases that can behave as a predicate link according to the evidence from the URDU.KON-TB treebank include adjective phrase, question adjective phrase, adverb phrase, clause, coordination conjunction phrase, case phrase, noun phrase, question noun phrase, preposition/postposition phrase, quantifier phrase, subordinating clause, vala phrase and verb infinitive phrase. The case of a PLINK can be a nominative or a possessive. The possible combinations of a PLINK with 1st division of functional tags e.g. spatial, temporal, instrumental, etc., can be seen in Table 3.7.

2.3.2.10 Modifier Or Adjunct

A modifier or an adjunct can be a word, phrase, or a clause that is behaving as an optional element or have a secondary importance in a sentence (Brinton and Brinton, 2010; Butt et al., 1999). The modifier is annotated with -MODF tag in our annotation. The examples can be seen in 2.72 and 2.71(a). The phrases that can behave as a modifier according to the evidences from the URDU.KON-TB treebank include adjective phrase, question adjective phrase, adverb phrase, question adverb phrase,

clause, coordination conjunction phrase, date phrase, case phrase, noun phrase, preposition/postposition phrase, subordinating clause, heading phrase, complex verb phrase, verb infinitive phrase and verb phrase. The case of a MODF can be a nominative or a possessive. The possible combinations of a MODF with 1st division of functional tags e.g. spatial, temporal, instrumental, etc., can be seen in Table 3.7.

This section concluded the functional annotation of the URDU.KON-TB treebank. All the subsections contained annotated examples for each functional category. The functional annotation is divided into two divisions. The first division contained the semantic tags which are shared at the POS and syntactic level of annotation, while the second division contained the functional tags. The semantic features include temporal, spatial, instrumental, etc. and the functional features include subject, object, oblique, etc. along with the labels for clauses and the anaphora resolution.

2.4 Summary

This chapter detailed an up to date annotation guideline for the URDU.KON-TB treebank. It included instructions for part of speech, syntactic and functional annotations. Twenty two main POS tags categories displayed in a Table 3.1 are divided into morphological and semantical subcategories. All of the possible combinations achieved are depicted in Table 2.1 of the semi-semantic POS tag set. The discussion of the tags along with the examples has been concluded in Section 2.1. Similarly, twenty six main syntactic tags displayed in the Table 3.4 were expanded hierarchically and thirty eight syntactic tags given in Table 2.2 have been discussed along with their examples in Section 2.2. The semantic features which can travel from lexical level to syntactic level are not discussed in Section 2.2, but in Section 2.3. This section concluded the functional annotation of the URDU.KON-TB treebank along with the examples. All the subsections contained annotated examples for each functional category. The functional tags in Table 2.3 are divided into two divisions. The first division is named as the semantic division. The second division is named as the functional division. The semantic features include temporal, spatial, instrumental, etc. and the functional features include subject, object, oblique, etc. along with some labels for clauses and anaphora resolution.

3

The URDU.KON-TB Treebank

This chapter details the development of a treebank for the South Asian language Urdu. Urdu is a comparatively under resourced language and the development of a treebank for Urdu will have significant impact on the state-of-the-art for Urdu language processing. In the URDU.KON-TB treebank described here, a semi-semantic part of speech tagset, a semi-semantic syntactic tagset and a functional tagset are proposed. The construction of the treebank is based on a corpus which is collected from Urdu Wikipedia and news papers. The sentences were annotated manually to ensure a high annotational quality. A hierarchical annotation[1] has a combination of a phrase structure, dependency structure and a hybrid dependency structure.

3.1 Background

A treebank or parsed corpus is a text corpus of sentences annotated with a syntactic structure (a tree structure), hence the name treebank. Similarly, corpus annotation is simply the process of the addition of interpretative linguistic information to a corpus. This process includes an addition of tags or labels in a text. The tags identify the class of words known as part of speech (POS) tagging, the arrangement of words and phrases to form a sentence is known as syntactic tagging (Leech, 2005). The difficulty of the task and the length of time required to build a treebank depends on the level of annotation detail and the breadth of the linguistic sample.

Phrase structures (PS) and dependency structures (DS) are the two traditional annotation structures for treebank annotation. A simple but large difference between

[1]During the development, an early version of the work (Abbas, 2012) was published, which is repeated here in this chapter with final updates.

phrase and dependency annotation structure is that in phrase structure annotation, the nodes represents phrases or constituents only e.g. a noun phrase NP and a verb phrase VP as shown in Figure 3.1(a) for a sentence *Michael loves Sara*, while in a dependency structure the nodes represent head words (with or without its syntactic tag) of its dependent words as shown in Figure 3.1(b) for the same sentence. The PS annotation was first introduced by Chomsky (1956) and its main focus is on building small simple constituents or phrases from the words of a sentence as shown in Figure 3.1(a) already. The DS annotation has its focus on building relations between the head word and its dependents. It was first proposed by Tesnière, Lucien. After his death, this theory of dependency structure was presented in (Tesnière and Fourquet, 1959). It is a common argument that the DS is not limited to a specific word order and well suited to free word order languages.

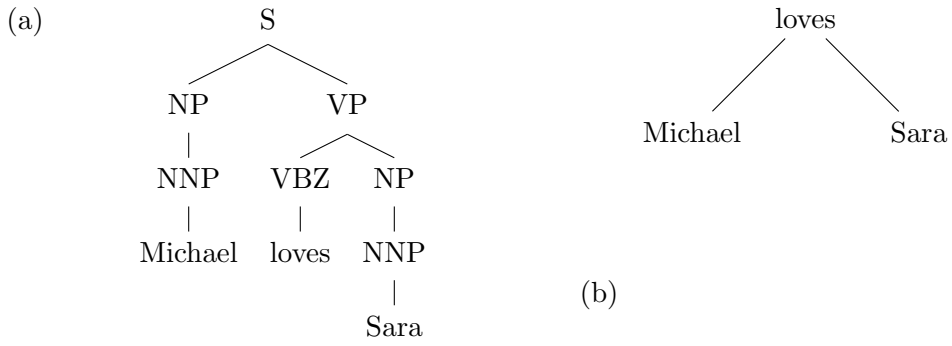


Figure 3.1: Phrase and dependency structure example

Some treebanks along with their annotation structure are as follows: the BulTreeBank[1] for the Bulgarian language follows HPSG (Head-driven Phrase Structure Grammar), the Penn Treebank[2] and ICE-GB[3] (International Corpus of the English-Great Britain) for English adopted the phrase structure scheme, the Prague Dependency Treebank[4] for the Czech language and the Quranic Arabic Dependency Treebank[5] for the Arabic language by University of Leeds, UK had adopted the dependency structure during treebank annotation. Almost 64 treebanks exist for different languages in the

[1] `http://www.bultreebank.org/` The project was funded by theVolkswagen Stiftung, Germany under the Programm "Cooperation with Natural and Engineering Scientists in Central and Eastern Europe".

[2] `http://www.cis.upenn.edu/~treebank/`

[3] `http://www.ucl.ac.uk/english-usage/projects/ice-gb/index.htm`

[4] `http://ufal.mff.cuni.cz/pdt/`

[5] `http://corpus.quran.com/`

world. At present, the languages for which more than three treebanks are available include Arabic, English, German, Italian, Latin and Japanese. A list of treebanks is given in Wallis (2008). The most popular treebank is the Penn treebank for the English language. A short description of some treebanks is given below and related work is discussed in Section 3.2.

The Penn Treebank for English: An annotated corpus containing 4.5 million words of American English collected from the Brown corpus. In the first phase of this project, POS information (Santorini, 1990) was encoded along with annotated syntactic structure in parallel with half of the corpus. A much work relies on the Penn Treebank e.g. stochastic parsing (Brill, 1991; Brill et al., 1990; Magerman and Marcus, 1990), skeletal parsing (Pereira and Schabes, 1992; Weischedel et al., 1991), training POS taggers (Meteer et al., 1991), disambiguating spoken sentences (Veilleux and Ostendorf, 1993), linguistic theory and psychological modeling (Niv, 1991), grammar development etc. The Penn Treebank had limitations like no clear argument/adjunct relationship, inconsistencies in the annotation scheme, limited annotation and need of predicate-argument structure. Other Treebanks for English are the Susanne Corpus (Sampson, 2002), the Lancaster Parsed Corpus (Leech, 1992), and International Corpus of English (Greenbaum, 1996). The German treebank TIGER (Brants et al., 2002) is based on the NEGRA treebank. The most popular treebank for German is the Tüba-D/Z. These treebanks (TIGER & Tüba-D/Z) contain 50000 and 85358 sentences, respectively, collected from German newspapers and also annotated with phrase and dependency structure. Each treebank used the Stuttgart Tübingen POS Tagset along with 49 and 36 grammatical function labels respectively (Schiller et al., 1995). TIGER has a flat annotation scheme with no unary branching while the other one allows for this and contains a deeper hierarchical structure.

POS tagging schemes are generally useful for differentiating words which have same spelling but different meanings, for example the word "present" may denote a noun *gift*, a verb to give someone a *present* or an adjective *not absent*. So, having a reliable and linguistically informed annotation scheme is very important. Moreover, there are many annotation schemes for corpora which include semantic (meanings of words), discourse (adding a information about anaphoric links), stylistic (adding information about speech and thought presentation), lexical (adding the identity of the lemma/base/stem of each word form in a text), etc. (Garside et al., 1997). Annotation can help automatic processing and analysis in many different ways. For example POS tagged corpora can generate a frequency list or frequency dictionaries with grammatical classification e.g. the verb "leaves" and the noun "leaves" should be treated differently

based on their frequency.

```
( S
      ( NP
            ( NNP Michael )
      )
      ( VP
            ( VBZ loves )
            ( NP
                  ( NNP Sara )
            )
      )
)
```

Figure 3.2: Bracket form of Figure 3.1(a).

A Penn treebank phrase structure example has already been depicted in Figure 3.1(a). In this tree, *Michael*, *loves* and *Sara* are the tokens of the sentence, which are labelled with POS tags as NNP (proper noun, singular), VBZ (verb, 3rd person, singular) and NNP respectively. The syntactic tags S, NP and VP represent the root of the sentence, a noun phrase and a verb phrase respectively. This phrase structure annotation scheme has the following advantages e.g. simplicity, easily convertible into bracketed sentences, 'light' on resources and the tree structure is relatively easy to read without specialized software tools. The leaf nodes represent tokens and the nodes connected directly to tokens represent POS tags of the respective tokens. Other nodes up above the POS tags represent the syntactic annotation for a sentence S in phrase structure. We have adopted the same phrase structure annotation along with the dependency and hybrid dependency structures (Section 3.3.2). The hierarchical relationships expressed by the trees are rendered via bracketing in computational treebanks. This can be done by transferring the human readable manually annotated trees from paper to machines. The machine readable bracketed form of Figure 3.1(a) is given in another version in Figure 3.2. The S in this bracketed form has the same two child nodes NP and VP as given in the tree representation. Similarly, the VP has the same structure of VBZ and NP at parallel level and so on. So, the difference is only easily human readable verses easily machine readable. Both the annotations have no any functional information e.g. grammatical, semantic or thematic, which is attempted in the development of the URDU.KON-TB treebank along with POS and syntactic an-

notation. The discussion of treebanks in this section is mainly not for Urdu language. Some efforts have been made in this regard, which are going to be discussed in Section 3.2.

3.2 Related Work

The selection of an annotation scheme is dependent on the constituent ordering of the language. Phrase structure is good for fixed constituent order languages like English, while dependency structure is good for free constituent order languages like Urdu, Hindi, German, etc. (Skut et al., 1997). The current work builds on a previous study at constructing a NU-FAST treebank (Abbas et al., 2009). However, the design of that treebank proved to be too simple and flat. It neither contained detailed morphological, syntactical and functional information, nor any information about displaced constituents or empty arguments. In addition, it was based on a POS tagset that is currently being revised by the author due to issues like ergative case marker *'nE'* in Figure 3.3, which should be tagged as case marker CM instead of particle P (Butt and King, 2004), etc. A tree for a sentence in 3.1 from the NU-FAST Treebank is given in Figure 3.3.

(3.1) حامد نے شیر کو افریقہ کے جنگل میں بندوق سے مارا ۔

hAmed=nE	*SEr=kO*	*afrIqah=kE*	*jangal=mEN*
Hamed.masc.sg=ERG	lion.masc.sg=ACC	Africa=POSS	jungle=SPT
bandUq=sE	*mArA*	.	
gun=INST	kill.masc.perf.		

'Hamid killed the lion with a gun in the jungle of Africa.'

This tree gives an idea of simplicity of NU-FAST treebank for Urdu. KP (case phrase), VP and NP are already known from previous discussions. The rest of the tags used in this sentence are PN for proper noun, P for particles including case markers, NN for noun, GP for genitive phrase, PRE for handling of prepositions, postpositions, etc. and similarly, PREP for preposition phrase even this is the case of postposition here in this figure, SEP for sE phrase even sE can be a instrumental case marker or a comparative postposition and VB deals all kinds of verbs.

Another Hindi-Urdu tree-banking (HUTB) effort is under way in a collaborative project[1] between five different universities (University of Colorado Boulder, Columbia

[1] `http://verbs.colorado.edu/hindiurdu/`

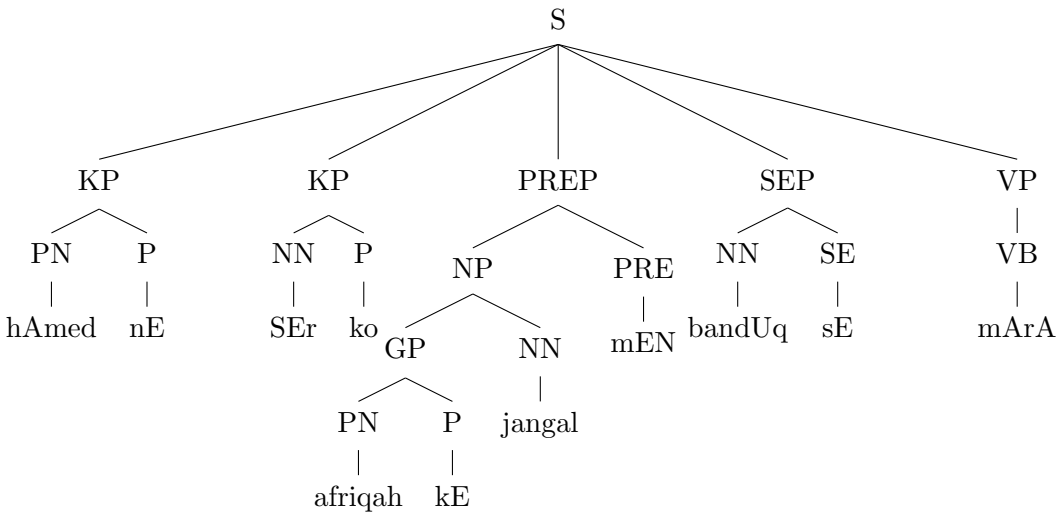


Figure 3.3: A sample tree for a sentence from NU-FAST Treebank

University, University of Massachusetts at Amherst, University of Washington, International Institute of Information Technology in India). However, the Urdu treebank being developed is comparatively small and is being done as part of a larger effort at establishing a treebank for Hindi. The beginning work of Hindi treebank was published in (Bhatt et al., 2009) and (Palmer et al., 2009). Although Urdu and Hindi share many structural features, there are some interesting differences and as the main effort of the Hyderabad-Colorado cooperation is focused on Hindi, many of the issues with respect to Urdu are not quite resolved. Our work takes up these issues and proposes solutions for them.

This Hindi-Urdu Treebank (HUTB) is multi-representational and multi-layered. Multi-layered is defined as the design of syntactic and lexical semantic annotation e.g. English PropBank (Palmer et al., 2005). Multi-representational is defined as the representation of choices e.g. in a sentence *Who do you think will come?*, *Who* can be an argument of *come* or not. These choices can be represented by crossing arcs to denote discontinuous constituent or a trace and co-indexation can be used. The HUTB contains three different annotations e.g. phrase structure annotation, dependency structure annotation and PropBank annotation. The guidelines are available on the portal mentioned earlier. The phrase structure (PS) adopted is based on the principles advocated by (Chomsky, 1993). The dependency structure (DS) is based on the Paninian grammatical model (Bharati et al., 1995, 2009b) and the PropBank

(PB) annotation structure is based on English PropBank (Palmer et al., 2005). The PropBank annotation deals with lexical predicate-argument structure. The DS and PB is annotated manually, while the PS is obtained automatically via using a conversion algorithm. The PS and DS representation is similar to annotations as depicted in Figures 3.1(a) and 3.1(b) respectively. The PropBank annotation labeling is different in that the predicate-arguments are annotated with labels as depicted in Figure 3.4 for the example sentence *A GM-Jaguar pact would give the U.S. car maker an eventual 30% stake in the British company.*

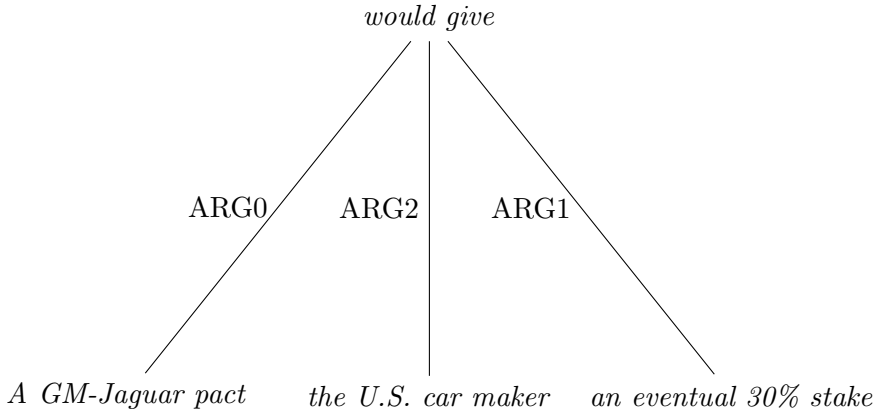


Figure 3.4: A PropBank example sentence

In this figure, the ARG0 is the *giver*, ARG1 is the *thing given* and ARG2 is the *entity given to*. In Hindi PropBank, these arguments are annotated by using karaka relations e.g. k1 to k5 according to the Paninian grammatical model (Butt, 2006). The DS+PB is enriched annotation and its conversion to PS builds a new resource using existing resources.

The output of the Urdu ParGram grammar project is also a kind of resource that lies in this domain of tree-banking proposed by Butt and King (2007). In the Urdu ParGram project, the built-in XLE[1] parser interface is used for the encoding of grammar. The parsed output of this grammar is the c-structure (syntactic tree) and f-structure (functional information). The Urdu part of the project is conducted at the University of Konstanz[2]. The modular architecture of LFG grammar project is explained in (Bögel et al., 2009). Initially, the aim of this project was only to develop grammars for different languages including Urdu. However, at present, the team of the ParGram project has added a new dimension towards the development of parallel treebanks (Sul-

[1] `http://www2.parc.com/isl/groups/nltt/xle/`

[2] `http://ling.uni-konstanz.de/pages/home/pargram_urdu/`

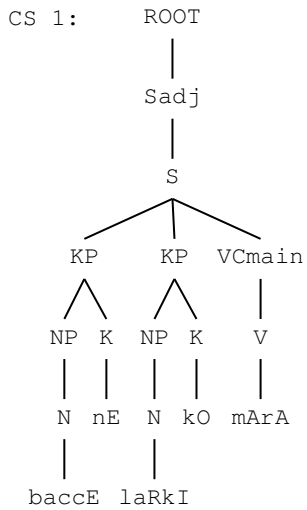


Figure 3.5: C-structure for a sentence

ger et al., 2013). The ParGram bank contains parallel treebanks for languages including English, Georgian, German, Hungarian, Indonesian, Norwegian, Polish, Turkish, Urdu and Wolof. Currently, fifty (50) sentences of each language are used for parsing on respective grammars, developed with the XLE interface. These sentences are translated from English into other language manually by native speakers. This output of the parsed sentences is collected in form of parallel corpora, i.e. a parallel treebank.

An output of the Urdu ParGram LFG grammar (Butt et al., 1999; Hautli et al., 2012) in the form of C and F structures produced after parsing the example sentence *baccE nE laRkI kO mArA* 'The child beat the girl' is given in Figures 3.5 and 3.6 respectively. The c-structure is a syntactic tree without predicate-argument information but the f-structure describes all the predicate-argument information e.g. subject, direct object, indirect object, etc. It also includes other functional information and morphological features. In short, the c and f structure represents a rich dependency annotation.

Today, many natural language processing (NLP) and machine learning (ML) applications rely on treebanks. Treebanks are heavily used in corpus linguistics for investigating syntactic phenomena or in computational linguistics for training or testing parsers. The value of parsed corpora is becoming more and more widely understood. Treebanks are important for syntactic research because of the high quality of linguis-

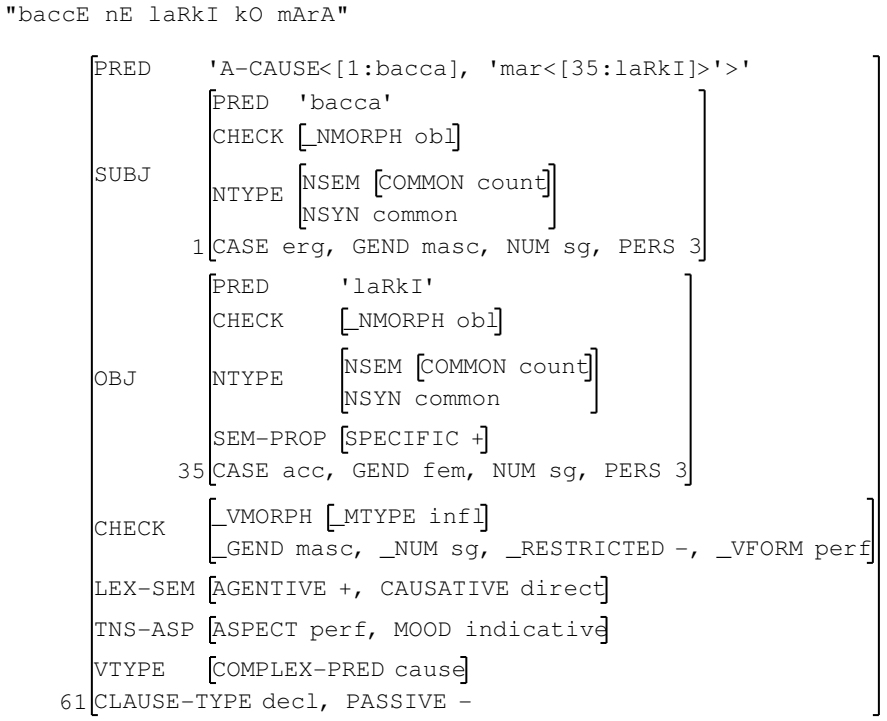


Figure 3.6: F-structure for a sentence

tics and empirical information that is provided. The overall state of NLP for Urdu is far behind that of European languages. So, there is much potential for industrial and academic work with respect to Urdu. This development of URDU.KON-TB treebank for Urdu will strengthen and boost linguists working in Urdu language processing.

3.3 The URDU.KON-TB Treebank

The development of the URDU.KON-TB treebank is performed in three steps, which includes the collection of sentences in the form of a corpus, manufacturing of an annotation scheme and the employment of annotation scheme on the said corpus. In initial development of the URDU.KON-TB treebank (Abbas, 2012), a POS, a syntactic and a functional tag sets were proposed, which are corrected and updated after annotation evaluation presented in Chapter 4. The updated versions of tag sets are presented and discussed here in this chapter in Section 3.3.2.

3.3.1 Corpus Construction

In our initial development (Abbas, 2012), we used a 19 million words corpus (Ijaz and Hussain, 2007) that was available at CRULP/CLE.[1] This corpus was collected from the Jang[2] and the BBC[3] newspapers. This corpus had licensing constraints due to which it is not publicly available anymore (Urooj et al., 2012). One thousand (1000) sentences taken from this corpus were then extensively modified to become free from licensing constraints, because we want to share our corpus freely under a Creative Commons Attribution/Share-Alike License 3.0 or higher. The next four hundred (400) sentences were collected from Urdu Wikipedia.[4] The data collected from Urdu Wikipedia is already under that license. Thus the size of the corpus is limited to fourteen hundred (1400) sentences. The size of corpus was kept limited within the context of this thesis. The corpus contains text of local & international news, social stories, sports, culture, finance, history, religion, traveling, etc.

3.3.2 Annotation Scheme

The annotation scheme of the URDU.KON-TB treebank consists of semi-semantic POS (SSP), semi-semantic syntactic (SSS) and functional (F) tag sets. The term semi-semantic (partly or partially semantic) is used with POS because some tags are encoded with semantics but not all e.g. N.SPT (a spatial noun) tag for a word *house*, ADJ.TMP (a temporal adjective) tag for a word *previous* in *previous year*, etc. The same concept is applied to SSS, when semantic information is mixed up with syntactic tags e.g. NP-SPT (a noun phrase having a spatial sense) tag for a phrase *a beautiful house*, NP-TMP (a noun phrase having a temporal sense) tag for a phrase *a long period*. At the POS level, a dot '.' is used to add morphological and semantical labelings of subcategories into the main categories as discussed in Section 3.3.2.1. At the constituent level, syntactic tags can be annotated with dot '.' or dash '-'. The dot is used here to represent syntactic features, while the dash is used to represent functional features, which includes semantical, clausal, grammatical and miscellaneous information. The details are given in Sections 3.3.2.2 and 3.3.2.3. A combination of phrase structure (PS) and hyper dependency structure (HDS) has been adopted. The DS is called HDS because it is not limited to make constituents on the basis of headwords, but also on the basis of the head-constituents, when you have to make a constituent from its nested

[1] A center for language engineering in Pakistan at `http://cle.org.pk/`

[2] `http://jang.com.pk/`

[3] `http://www.bbc.co.uk/urdu/`

[4] http://ur.wikipedia.org/wiki/صفحہ اول

constituents. The POS, morphological, syntactical, semantical, clausal and functional information all together, makes a rich annotation scheme for our Urdu treebank. The need for such type of schemes is highly advocated such as (Clark et al., 2010; Skut et al., 1997), etc.

3.3.2.1 Semi-Semantic POS (SSP) Tagset

A simple POS tag set was devised first, which contained twenty two (22) main POS-Tag categories displayed in Table 3.1. The description of the tags is given in the respective cells of the table. The tables includes some non-familiar tags like HADEES and MARKER to represent the Arabic statements of prophets in Urdu text and a phrase or a sentence marker similar to punctuation marks but not all, respectively. The labels for morphological and semantic subcategories are presented in Tables 3.2 and 3.6 respectively, which can be added to 22 main categories of POS tags by using a dot '.' symbol. The SSP tag set was refined during the manual annotation process of sentences and further refined after the evaluation process with Krippendorf's α statistical model discussed in Chapter 4 and also presented in (Abbas, 2014a). The final refined form of the SSP tag set is given in Table 3.3. In case of morphology, if a main verb V has a perfective morphology, then the tag becomes V.PERF. Similarly, the case of spatial noun N.SPT is discussed in the beginning of Section 3.3.2. The semantic tags like SPT, TMP, MNR, etc. are not possible with verbs, auxiliaries, conjunctions, etc. as can be seen in Table 3.3.

Table 3.1: The main POS-Tag categories for the URDU.KON-TB treebank

ADJ (Adjective)	PRAY (Specific statements of prayers)
ADV (Adverb)	PREP (Preposition)
C (Conjunction)	PT (Particle)
CM (Case marker)	Q (Quantifier)
DATE (Date)	QW (Question word)
HADEES (Narration of prophets deeds)	SYM (Symbol)
INT (Interjection)	TTL (Title)
M (Marker)	U (Unit)
N (Noun)	V (Verb)
P (Pronoun)	VALA (Special Word Vala)
POSTP (Postposition)	VAUX (Verb auxiliary)

(3.2) حامد نے شیر کو جنگل میں بندوق سے مارا ۔

hAmed	*nE*	*SEr*	*kO*	*jangal*	*mEN*	*bandUq*	*sE*	*mArA*	.
N.PROP	CM	N	CM	N.SPT	CM	N	CM	V.PERF	M.S

Table 3.2: Morphological tag set to annotate subcategories of verbs and auxiliaries

IMPERF (Imperfective form)	PROG (Progressive form)
PERF (Perfective form)	PASS (Passive form)
ROOT (Root form)	FUTR (Future tense)
SUBTV (Subjunctive form)	PAST (Past tense)
INF (Infinite form)	PRES (Present tense)

'Hamid killed the lion in the jungle with a gun.'

An example of SSP tag annotation is given in example 3.2. The Urdu script is written from right to left. The row beneath the Urdu script is the transliteration of the sentence as proposed in Malik et al. (2010). The tokens of the sentence are tagged according to the SSP tag set. *hAmed* is a proper name. *SEr* and *bandUq* are common nouns, while *jangal* is a spatial common noun.[1] *nE*, *kO*, *mEN*, and *sE* are case markers for ergative, accusative, spatial/locative and instrumental cases, respectively. The syntactic differentiation of the case markers is done according to the studies in Butt and King (2004).

The tagset in Table 3.3 represents the complete SSP tagset. The discussion on each tag can be found in the annotation guidelines of the URDU.KON-TB treebank presented in Chapter 2. As an example, consider the ADJ (Adjective) in Table 3.3, which is divided into five subcategories of tags DEG (Degree), ECO (Echo), MNR (Manner), SPT (Spatial) and TMP (Temporal). Relevant examples are provided in 3.3.

(3.3) (a) اچھا لڑکا 'Most important personality'

acHA *laRkA*
ADJ N
'Good boy'

(b) اہم ترین شخصیت

aham *tarIn* *Saxs2iat*
N ADJ.DEG ADJ

[1] In the presence of a sense of place/location or direction to/from place/location in a word, SPT tag is used e.g. *Pakistan* and the *country* are the two words. *Pakistan* is the proper name of a place (country) and is tagged as N.PROP.SPT. However, *country* is a common noun but having a sense of place. So, it is tagged as N.SPT. This distinction is not different from spatial adverbs e.g. *there*, *here*, etc.

Table 3.3: A detailed version of the SSP tagset for the URDU.KON-TB treebank

ADJ (Adjective)	.REL (Relative)	.ROOT (Root)
.DEG (Degree)	.DEM (Demons...)	.SUBTV (Subjunctive)
.ECO (Echo)	.PERS (Personal)	.PAST (Past)
.MNR (Manner)	POSTP (Postposition)	.PRES (Present)
.SPT (Spatial)	.CMP (Comparative)	.LIGHTV (Light Verb)
.TMP (Temporal)	.MNR (Manner)	.IMPERF (Imperfective)
ADV (Adverb)	.POSS (Possessive)	.INF (Infinite)
.DEG (Degree)	.REP (Repeat)	.PERF (Perfective)
.MNR (Manner)	.SPT (Spatial)	.ROOT (Root)
.NEG (Negative)	.TMP (Temporal)	.SUBTV (Subjunctive)
.SPT (Spatial)	PRAY (Pray)	.MOD (Modal)
.TMP (Temporal)	PREP (Preposition)	.IMPERF (Imperfective)
.REL (Relative)	.MNR (Manner)	.PERF (Perfective)
C (Conjunction)	.SPT (Spatial)	.SUBTV (Subjunctive)
.CAUS (Causative)	.TMP (Temporal)	.PERF (Perfective)
.CONS (Concessive)	PT (Particle)	.REP (Repeat)
.CORD (Coordinative)	.ADJ (Adjective)	.ROOT (Root)
.CORR (Co-relative)	.EMP (Emphatic)	.REP (Repeat)
.SBORD (Subordinating)	.INTF (Intensifier)	.SUBTV (Subjunctive)
.COND (Conditional)	.RESULT (Result)	.PAST (Past)
CM (Case Marker)	Q (Quantifier)	.PRES (Present)
DATE (Date)	.ADJ (Adjective)	VALA (Vala)
.D (Day)	.CARD (Cardinal)	VAUX (Verb Auxiliary)
.M (Month)	.FRAC (Fractional)	.IMPERF (Imperfective)
.Y (Year)	.ORD (Ordinal)	.INF (Infinite)
HADEES (Hadees)	QW (Question Word)	.MOD (Modal)
INT (Interjection)	.REP (Repeat)	.IMPERF (Imperfective)
M (Marker)	.TMP (Temporal)	.PERF (Perfective)
.P (Phrase)	.SPT (Spatial)	.SUBTV (Subjunctive)
.S (Sentence)	.MNR (Manner)	.PASS (Passive)
N (Noun)	SYM (Symbol)	.IMPERF (Imperfective)
.ADJ (Adjective)	TTL (Title)	.INF (Infinite)
.MNR (Manner)	.REG (Regard)	.PERF (Perfective)
.REP (Repeat)	U (Unit)	.ROOT (Root)
.PROP (Proper)	V (Verb)	.SUBTV (Subjunctive)
.SPT (Spatial)	.COP (Copula)	.PERF (Perfective)
.TMP (Temporal)	.IMPERF (Imperfective)	.PROG (Progressive)
.REP (Repeat)	.PERF (Perfective)	.ROOT (Root)
.SPT (Spatial)	.ROOT (Root)	.SUBTV (Subjunctive)
.REP (Repeat)	.SUBTV (Subjunctive)	.FUTR (Future)
.TMP (Temporal)	.PAST (Past)	.PAST (Past)
.REP (Repeat)	.PRES (Present)	.PRES (Present)
P (Pronoun)	.IMPERF (Imperfective)	
.DEM (Demonstrative)	.REP (Repeat)	
.INDF (Indefinite)	.INF (Infinite)	
.PERS (Personal)	.LIGHT (Light)	
.POSS (Possessive)	.IMPERF (Impe...)	
.REF (Reflexive)	.INF (Infinite)	
.REP (Repeat)	.PERF (Perfective)	
.REF (Reflexive)	.PROG (Progressive)	

(c) برا ورا کام

burA *vurA* *kAm*
ADJ ADJ.ECO N

'Ugly work'

(d) جابرانہ حکومت

jaberaanah *hakUmat*
ADJ.MNR N

'Forceful government'

(e) گزشتہ سال

guzaStah *sAl*
ADJ.TMP N
'Previous year'

(f) ملتانی کھسہ

mUltAnI *kHUsah*
ADJ.SPT N
'Multani shoe'

The example 3.3(a) is a simple case of ADJ, while 3.3(b) is a case of a degree adjective[1] annotated with ADJ.DEG. The comparative and superlative forms of adjectives can be made by introducing Persian suffixes *tar* 'more' and *tarIn* 'most' after the absolute form of adjectives e.g. *xUbs3Urat-tar* 'prettier' and *Ubs3Urat-tarIn* 'prettiest'. There are some words which can play the role of a degree adverb or a degree adjective e.g. *zEyAdah* 'more/most/much', *bohat* 'more/enough', *kAfI* 'quite/too', etc. (Schmidt, 2013). If these words qualify adjectives, then this is the usage as degree adverbs, otherwise as a degree adjective. Example 3.3(c) is a case of reduplication (Abbi, 1992; Bögel et al., 2007). As reduplication has two versions. First, in Urdu like other South Asian languages, the reduplication of a content word is frequent. Its effect is only to strengthen the proceeding word or to expand the specific idea of a proceeding word into a general form e.g. *kAm THIk-THAk karnA* 'Do the work right' or *kOI kapRE-vapRE dE dO* 'Give me the clothes or something like those'. Second version is the repetition of the original word e.g. *sAtH sAtH* 'with/along-with'. These two versions are named as *full word reduplication* and *echo reduplication* by Bögel et al. (2007), which are represented in our annotation as ECO (echo) and REP (repetition) respectively. The echo words normally begin with the letters *S* or *v* or *m*.

Example 3.3(d) is the case of adjective having a sense of manner annotated as ADJ.MNR. If an adjective qualifies a action noun, then a sense of action or something is produced, whose behavior or the way to do that action is confirmed through ADJ.MNR e.g. *z4AlemAnah t2abdIlIyAN* 'brutal changes'. If an adjective comes individually, then its mannerism can be resolved independently through its sense or by building a sense with the predicate. If there is a sense of manner then an adjective of manner can exist like in copular construction e.g. *voh GEr-h2Az3ir hE* 'He is absent'. An exercise of adjectives and adverbs of manner for the English language can be seen at Cambridge University.[2] Example 3.3(e) is case of an adjective having a temporal sense. Finally, example 3.3(f) is the case of an adjective having a spatial sense. The adjective used

[1]This division is used to represent absolute, comparative and superlative degree in adjectives and adverbs

[2]http://www.cambridge.org/grammarandbeyond/wp-content/uploads/2012/09/Communicative_Activity_Hi-BegIntermediate-Adjectives_and_Adverbs.pdf

here is the derivational form of a city/place name *Multan*, which is a spatial proper noun. But it appears here as an adjective and annotated as ADJ.SPT[1] like in this sentence e.g. *voh Ek pAkistAnI laRkA hE* 'He is a pakistani boy'.

The example 3.3 for adjectives exploited its POS tags along with semantic tagging like TMP, SPT, MNR, etc. However, to give an introduction about morphology and verb functions, another POS category V from Table 3.3 is discussed as follows. A few high quality studies were conducted on verbs for morphologically rich language (MRL) Urdu by Butt and Rizvi (2010), Butt and Ramchand (2001) and Butt (2010). The rules for identifying different forms of verbs were adopted from these studies. The V annotates the predicate/main-verb of the sentence and is divided mainly into 11 subcategories, which include COP (copula verb), IMPERF (imperfective morphological form of verb), INF (infinitive form of verb), LIGHT (1st light verb with nouns and adjectives), LIGHTV (2nd light verb with verbs), MOD (modal verb), PERF (perfective morphology), ROOT (root form), SUBTV (subjunctive form), PAST (past tense of a verb) and PRES (present tense of a verb). Their description is also given in Table 3.3. These tags are further divided into subcategories depicted in Table 3.3. All these tags represents different morphological forms and the function of a verb that it governs. Some annotated sentences containing different verb forms and functions are given in example 3.4.

(3.4) (a) مہنگائی نے لوگوں کا جینا دوبھر کیا تھا

mehangAI	*nE*	*lOgON*	*kA*	*jInA*	*dUbHar*	*kiyA*
N	CM	N	CM	N	N	V.LIGHT.PERF
tHA						
VAUX.PAST						

'The inflation had made the life of people hard'

(b) گرانفروشوں کے خلاف قانون حرکت میں لایا جائے

giraN-faroSoN	*kE*	*xilAf*	*qAnUn*	*harkat*	*mEN*	*lAyA*
N	CM	POSTP.MNR	N	N	CM	V.PERF
jAyE						
VAUX.PASS.SUBTV						

'The law should be practiced against inflators'

[1]Spatial adjectives are used to describe a place/location, direction or distance e.g. multAnI 'Multani', aglI 'next', and dUr 'far' respectively

(c) محمد صلی اللّٰہ علیہ والہٖ وسلم نے فرمایا کہ الحسین منی و انا من الحسین
یعنی حسین مجھ سے ہے اور میں حسین سے ہوں .

mUhammad *sal-lal-la-ho-a2lEhE-va-AlEhI-salam* *nE* *farmAyA*
N.PROP PRAY CM V.PERF
keh *“* *al-hUsynON-mInnI-vA-anA-mInal-hUsyn* *”* *ya2nI*
C.SBORD M.P HADEES M.P ADV
‘ *hUsyn* *mUjH* *sE* *hE* *aOr* *mEN*
M.P N.PROP P.PERS CM V.COP.PRES C.CORD P.PERS
hUsyn *sE* *hUN* *’* *.*
N.PROP CM V.SUBTV M.P M.S
‘Muhammad (May Allah grant peace and honor on him and his family) said that “al-hUsynON-mInnI-vA-anA-mInal-hUsyn” means ‘Hussain is from me and I am from Hussain’ . ’

(d) تم نے حج تو کر لیا ہو گا ؟

tUm *nE* *haj* *tO* *kar* *liyA*
P.PERS CM N PT.EMP V.ROOT V.LIGHTV.PERF
hO *gA* *?*
VAUX.SUBTV VAUX.FUTR M.S
‘You will have made the pilgrimage?’

(e) جب یہاں کھیت ہوتے تھے

jab *yahAN* *kHEt* *hotE* *tHE*
ADV.TMP.REL ADV.SPT N.SPT V.IMPERF VAUX.PAST
’When, here would have been crop fields’

(f) ان کا مطالبہ تھا

Un *kA* *mUtAlibah* *tHA*
P.PERS CM N V.PAST
‘It is their demand.’

The sentence in example 3.4(a) is a case of noun-verb complex verb predicate, which was first proposed by Mohanan (1994). The words *dUbHar kiyA* ‘made hard’ is a noun-verb complex predicate. The noun *dubHar* and the verb *kiyA* with a perfective morphological form *yA* at the end are annotated as a N and a V.LIGHT.PERF respectively. Similarly, a perfective verb *liyA* ‘took’ after a root form of verb *kar* ‘do’ is an example of the verb-verb complex predicate depicted in 3.4(d). This construction is adopted from the studies given in (Butt, 2010). The light verb after a N or an ADJ

lies in the 1st category of light verbs and annotated as V.LIGHT in our annotation, while the light verb after a verb lies in the 2nd category of light verbs and annotated as V.LIGHTV. The next sentence in 3.4(b) is a passive sentence. A passive construction can be concluded with the inflected form of a verb jAnA 'to go' preceded by another verb with perfective morphology as can be seen in 3.4(b). The subjunctive form of auxiliary verb tagged as VAUX.PASS.SUBTV is preceded by a perfective verb *lAyA* 'brought', which is then annotated as V.PERF. The subjunctive form of verb is acting as an aspectual auxiliary and not as a V.LIGHTV, which was discussed in (Butt and Ramchand, 2001) and adopted as it is. The rules for identification of verb function and other morphological forms can be found in annotation guidelines in Chapter 2.

To explore some other unusual tags, a long sentence is presented in 3.4(c). After the name of prophets or righteous religious-personalities, some specific and limited prayers called *s3alAvAt* 'prayers' e.g. *sal-lal-la-ho-a2lEhE-va-AlEhI-salam* 'May Allah grant peace and honor on him and his family', *a2lEh salAm* 'peace be upon him', etc. in Arabic is most likely in Urdu text and annotated as PRAY. Similarly, the statements of prophet Muhammad (PBUH) called *h2adIs2* 'narration' e.g. *In-namal-aa2mAlo-bin-niyAt* 'The deeds are considered by the intensions' in Arabic is also a tradition in Urdu text and annotated as HADEES. In religious text of Urdu, this kind of phenomenon is most likely in Arabic script rather than the Urdu script. This annotation with PRAY and HADEES is performed only, when prayers or narrations appear in Arabic language in Urdu text as can be seen in 3.4(c). The phrase markers like comma, double quotes, single quotes, etc. are annotated with M.P and sentence marker like full-stop, question mark, etc. are annotated with M.S as presented in the same example. The tense is divided into present, past and future. A predicate of the sentence with present and past tense is possible as annotated in 3.4(f) but not with future tense, because future tense always behaves as verb auxiliary in Urdu. The tense of verb auxiliaries like present, past and future is annotated in 3.4(a,d,e). A verb with imperfective morphology e.g. *tA, tI, tE, tEN* at the end of a verb is annotated with V.IMPERF as given in 3.4(e).

This section introduced the concept of semi-semantic part of speech tags used in our annotation. There are twenty two tags, which are divided into further subcategories as presented in Table 3.3. The POS annotation evaluation via Krippendorf's Alpha α is discussed in Chapter 4, which came up with POS tags issues related to readability. After evaluation, the problematic POS tags are either removed or revised and a final SSP tag is obtained as discussed.

3.3.2.2 Semi-semantic Syntactic Tagset

This section presents the syntactic tag set SSS for the URDU.KON-TB treebank. The tag set is named as semi-semantic syntactic (SSS) tag set because the main syntactic categories displayed in Table 3.4 can be added with semantic labels presented in Table 3.6. After developing this tag set, the POS tagged sentences of the corpus were annotated. During manual annotation of sentences, the tag set was refined and this refinement process continued until the last phase of annotation evaluation. In the last phase, the revisions were made during the training course of annotation evaluation and finally, the reliability of the SSS annotation was evaluated via Krippendorf's α statistical model, see Chapter 4. There are twenty six (26) main syntactic tag categories, which are displayed in Table 3.4. The description of the tags is given in the respective cells of the table. These main categories are further divided into syntactical and semantical subcategories given in Table 3.5. The dot '.' is used to add further syntactical subcategories, while the dash '-' is used to add semantical subcategories to a main category e.g. in NP.NOM-TMP, the main syntactic category of noun phrase NP with a nominative NOM case has a temporal semantics.

Table 3.4: The main syntactic-tag categories for the URDU.KON-TB treebank.

ADJP (Adjective Phrase)	QP (Quantifier Phrase)
ADJPQ (Question Adjective Phrase)	QWP (Question Word Phrase)
ADVP (Adverb Phrase)	S (Sentence)
ADVPQ (Question Adverb Phrase)	SBAR (Subordinate Clause)
CL (Clause in a sentence)	SBARQ (Question Subordinate Clause)
CORDP (Coordination Phrase)	SQ (Question Sentence)
DATEP (Date Phrase)	HP (Heading Phrase)
PARP (Parenthetic Phrase)	UP (Unit Phrase)
KP (Case Phrase)	VALAP (Vala Phrase)
KPQ (Question Case Phrase)	VCMAIN (Main Verb Phrase)
NP (Noun Phrase)	VCP (Verb Complex Phrase)
NPQ (Question Noun Phrase)	VIP (Verb Infinite Phrase)
PP (Pre/Post-position Phrase)	VP (Verb Phrase)

The basic syntactic tags like ADJP, ADVP, NP, PP, SBAR, etc. are adopted from Penn treebank along with some new tags like ADJPQ, ADVPQ, KP, QWP, VCP, VIP, etc. These main categories are divided into their syntactical and semantical sub categories, which are depicted into a final and detailed version of SSS tag set in Table 3.5. Similar to the SSP tag set depicted in Table 3.3, this SSS tag set also has a same hierarchical structure.

Table 3.5: A detailed version of SSS tagset for the URDU.KON-TB treebank

ADJP (Adjective Phrase)
-MNR (having semantics of manner)
-SPT (having spatial semantics)
-TMP (having a temporal semantics)
ADJPQ (adjective phrase contain question)
ADVP (adverb phrase)
-MNR (having semantics of manner)
-SPT (having spatial semantics)
-TMP (having a temporal semantics)
ADVPQ (adverb phrase contain question)
-MNR (having semantics of manner)
-TMP (having a temporal semantics)
CL (clause)
CORDP (coordination phrase)
.NOM (has a nominative case)
-SPT (having spatial semantics)
-INST (has an instrumental case)
-MNR (having semantics of manner)
-SPT (having spatial semantics)
-TMP (having a temporal semantics)
DATEP (date phrase)
-TMP (having a temporal semantics)
PARP (Parenthetic phrase)
KP (case phrase)
.ACC (has an accusative case)
-SPT (having spatial semantics)
.DAT (has a dative case)
-SPT (having a spatial semantics)
.ERG (has an ergative case)
-SPT (having spatial semantics)
.POSS (has a possessive case)
-MNR (having semantics of manner)
-SPT (having a spatial semantics)
-TMP (having a temporal semantics)
-INST (has an instrumental case)
-MNR (having semantics of manner)
-SPT (having a spatial semantics)
-TMP (having a temporal semantics)
-MNR (having semantics of manner)
-SPT (having a spatial semantics)
-TMP (having a temporal semantics)
KPQ (case phrase contain a question)
.POSS (has a possessive case)
-INST (has an instrumental case)
NP (noun phrase)
.ACC (has an accusative case)
.DAT (has a dative case)
.NOM (has a nominative case)
-MNR (having semantics of manner)
-SPT (having a spatial semantics)
-TMP (having a temporal semantics)
.POSS (has a possessive case)
-MNR (having semantics of manner)
-SPT (having a spatial semantics)
-TMP (having a temporal semantics)
NPQ (noun phrase contain a question)
.ACC (has an accusative case)
.NOM (has a nominative case)
PP (pre/post-position phrase)
-CMP (has a semantics of comparison)
-SPT (having a spatial semantics)
-TMP (having a temporal semantics)
-INST (has an instrumental case)
-MNR (having semantics of manner)
-POSS (has a possessive case)
-SPT (having a spatial semantics)
-TMP (having a temporal semantics)
QP (quantifier phrase)
QWP (question word phrase)
S (root of a sentence structure)
SBAR (subordinate clause in a sentence)
-CORR (correlative clause)
-REL (relative clause)
-RESULT (resultant clause)
SBARQ (subordinate clause contain question)
-RESULT (resultant clause)
SQ (yes/no question sentence)
HP (heading phrase)
UP (unit phrase)
VALAP (Vala phrase)
VCMAIN (main verb of a sentence)
VCP (verb phrase contain complex predicate)
VIP (verb infinitive phrase)
VP (verb phrase)

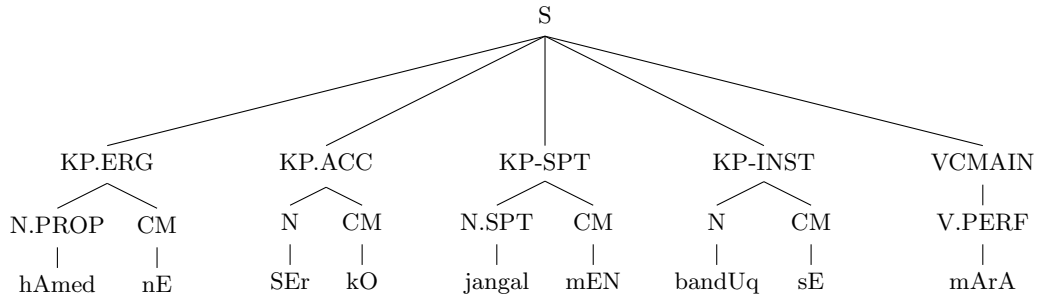
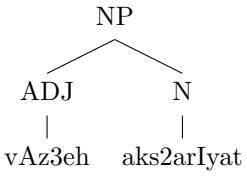


Figure 3.7: A SSS annotation of example 3.2 for KP subcategories

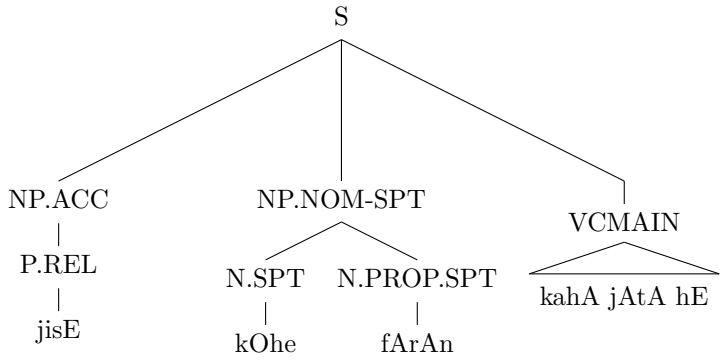
The SSS annotation of example 3.2 is given in Figure 3.7. The tags with dash '-' are semantic labels given in Table 3.6. The case phrase KP is built, when a case marker becomes the head word in a phrase. These case markers include *kA, kI, kE, kO, nE, mEN, par, tak* and *sE*. The *kA, kI, kE* 'of' are the possessive case markers. The *kO* 'to' can be accusative or dative, *nE* is an ergative case marker, while *mEN, par, tak* are locative or spatial case markers. The final *sE* is an instrumental case marker. The detailed study of case markers can be seen in (Butt, 2006). The KP is divided into four syntactic subcategories along with semantical subcategories, which include KP.ACC (KP with an accusative case), KP.DAT (KP with a dative case), KP.ERG (KP with an ergative case) and KP.POSS (KP with a possessive case). In Figure 3.7 from left to right, the first phrase becomes KP due to head word CM. The form of the case marker is ERG for ergative CM *nE*, which is then added to KP. An ergative CM identifies the subject of the sentence, which will be discussed in 3.3.2.3. The second CM from left to right in the tree is the head word *kO* 'to' that can identify accusative and dative objects. In this sentence, there is no indirect object, therefore the KP has an accusative case ACC, which is projected at syntactic level. The third CM is a head word *mEN* 'in/into', which is itself a spatial CM. In the KP annotation, when the dependent word or head word has some semantic labeling, then this semantic property is projected at syntactic level if the head word is a CM. As the spatial noun N.SPT has a spatial semantics, so it is projected at syntactic level by using a dash '-' after KP, which is acting as a modifier of this sentence. The last CM is *sE* 'with/from', which can be instrumental, source expression, locative, temporal expression, material, comparative postposition, etc (Mohanan, 1994; Raza, 2011). The instrumental noun *bandUq* adds a INST semantics to last KP phrase, which is acting as a modifier of the sentence. The VCMAIN is the main verb phrase of the sentence adopted from Urdu LFG grammar

project (Butt and King, 2007). The detail of case marking for Urdu language can be seen in Butt and King (2004), Butt and Ahmed (2011) and Khan (2011).

(a). *vAz3eh aks2arIyat* 'clear majority'



(b). *jisE kOhe fArAn kahA jAtA hE* 'Which is called the mountain of Faran'



(c). *mujHE mErE dE dO* 'Give me mines'

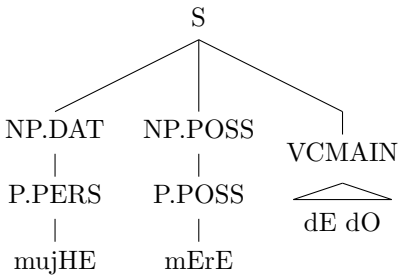


Figure 3.8: A SSS annotation of NP and its subcategories

In Figure 3.7, the grammatical labeling like subject, object, etc. is not annotated, which will be discussed and annotated in related Section 3.3.2.3. In SSS annotation, the noun phrase NP is built, when a noun becomes the head word. An NP is divided into four syntactic subcategories along with the semantical subcategories, which include NP.ACC (noun phrase has an accusative case), NP.DAT (noun phrase has a dative case), NP.NOM (noun phrase has a nominative case) and NP.POSS (noun phrase has a possessive case). These four subcategories along with the major syntactic category NP and semantical subcategories conclude 11 divisions presented in Table 3.5. Some of its major annotated examples can be seen in Figure 3.8. A noun phrase in Figure 3.8(a) is

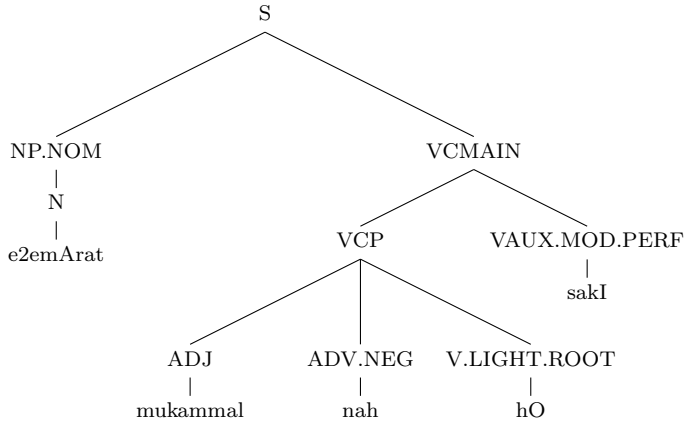
the simple case of NP, in which an adjective ADJ is the modifier of a head word N. The annotated sentence in Figure 3.8(b) is the case of passive construction similar to Figure 3.4(b). A relative pronoun *jisE* 'which' annotated as P.REL is an inflected form of *jis kO* and same is the case with personal pronoun *mujHE* 'me' annotated as P.PERS in Figure 3.8(c). In an active sentence, if the case marker *kO* comes with the subject, then it is the case of dative case marker or dative subject as annotated with NP.DAT in Figure 3.8(c) (Butt and King, 2004). However, in passive sentence, the case marker *kO* with the subject identifies an accusative case given in Figure 3.8(b) (Tiwari, 2004). In Figure 3.8(b), a phrase NP.NOM-SPT is formed due to a spatial proper noun *fArAn*, which is acting as a head word. Similarly, in Figure 3.8(c), a phrase NP.POSS is built due to a possessive pronoun *mErE* 'mine' and its possessive property is projected at syntactic level.

The tree in Figure 3.9(a) is the case of verb complex predicate phrase VCP (Butt, 2010) and modal verbs (Abbas and Nabi Khan, 2009; Bhatt et al., 2011). A phrase VCP is constructed when a light verb annotated with LIGHT or LIGHTV becomes the head word. The VCP in figure has a combination of an adjective *mukammal* 'complete' annotated with ADJ and a light verb *hO* 'be/become' annotated as V.LIGHT.ROOT with root form. The negative adverb annotated with ADV.NEG like *nah* 'not' and *nahIN* 'not' can be the part of this complex predicate combination. Due to the light verb, the phrase becomes VCP, which is in parallel to a modal verb *sakI* 'could' annotated as VAUX.MOD.PERF with perfective morphology in VCMAIN. Figure 3.9(b) is the simple case of adverb phrase ADVP, in which an adverb *fIs3ad* 'percent' annotated with ADV is acting as a head word.

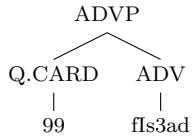
In Figure 3.9(c), an adverb phrase annotated with ADVP-MNR is formed because the head word *is-t2arah2* 'like this/in this way' is an adverb annotated with ADV.MNR having a semantics of manner (Schmidt, 2013). This semantic property of head word is then projected at syntactic level according to annotation guidelines given in Chapter 2. Figure 3.9(d) is the collective example of temporal and spatial adverbs *jab* 'when' and *yahAn* 'here' annotated with ADV.TMP and ADV.SPT respectively. Their definitions by Schmidt (2013) were discussed in Section 3.3.2.1 along with adjectives. Both temporal and spatial adverb are acting as head words in their respective phrases and the semantic projection at syntactic level annotated with ADVP-TMP and ADVP-SPT respectively is similar to Figure 3.9(c).

As a last example of this section, two sentences are syntactically annotated and presented in Figure 3.10. The tree presented in Figure 3.10(a) discussed the genitive or possessive case marker (Bögel and Butt, 2013; Butt and King, 2003) and the verb-verb

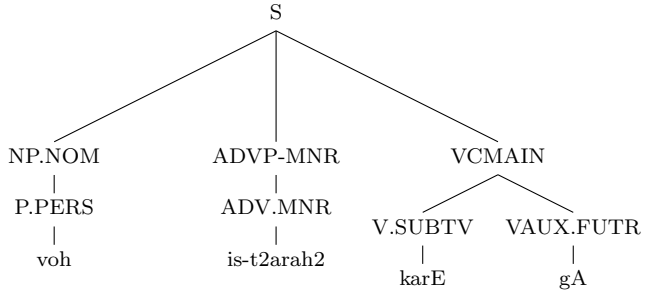
(a). *e2emArat mukammal nah hO saki* 'The building could not be completed'



(b). *99 fIs3ad* '99 percent'



(c). *voh is-t2arah2 karE gA* 'He will do like this'



(d). *jab yahAN gHar hotE tHE* 'When, here would have been houses'

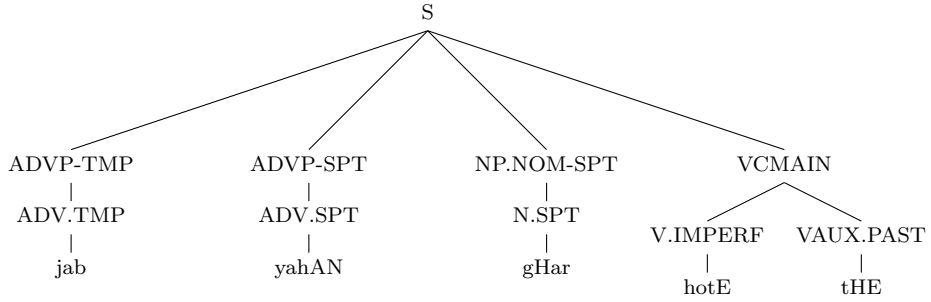


Figure 3.9: A SSS annotation of VCP and ADVP with subcategories

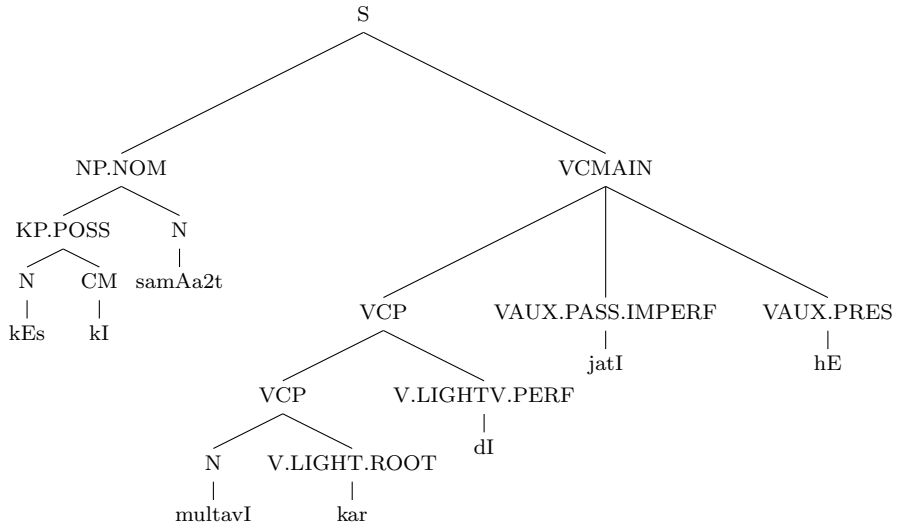
complex predicate i.e. a light verb after a verb (Butt, 2010). The *kA, kI, kE* 'of' are the possessive case markers. A phrase KP.POSS is formed, when a possessive case marker becomes the head word e.g. in *kEs kI* 'of case', a word *kI* 'of' is the head word causing the construction of KP.POSS phrase in the tree. A phrase VCP is built, when a light verb becomes the head word. In VCMAIN of Figure 3.10(a), there are two phrases of VCP formed due to two different subcategories of complex predicates annotated as V.LIGHT.ROOT and V.LIGHTV.PERF respectively. The VCP formed due to V.LIGHT.ROOT in *multavI kar* 'do adjournment' is the case of noun-verb complex predicate discovered by Ahmed and Butt (2011) and the VCP formed in *multavI kar dI* 'did adjournment' due to *dI* 'gave' annotated as V.LIGHTV.PERF after a root form of verb *kar* 'do' is the case of verb-verb complex predicate invented by Butt (1995). The rules for identifying complex predicates are presented at the end of Section 4.4.1 and its annotation guidelines in Chapter 2. Figure 3.10(b) contains a pre/post-position phrase PP, which can be formed, when a preposition or a postposition becomes the head word. The PP contained a KP.POSS phrase formed due to a possessive case marker *kE* 'of' behaving as a head word in *giraN-faroSoN kE* 'of inflators'. After the construction of KP.POSS, a postposition *xilAf* 'against' annotated as POSTP.MNR is acting as the head word and causing the construction of PP with projected semantic property of MNR. The rest of the phrases in this annotated tree are already discussed in earlier examples.

To summarize, the main syntactic tag set given in Figure 3.4 consists of twenty six tags. These tags are divided into further syntactical and semantical sub categories presented in Table 3.5. After getting observation from annotators during training course of evaluation discussed in Chapter 4, the SSS tags are refined and presented in Table 3.5. The refinement is the process of keeping or throwing the tags from the SSS tag set. The inter-annotator agreement or reliability of SSS annotation is evaluated through Krippendorf's α coefficient, which is also presented in Chapter 4.

3.3.2.3 Functional Tagset

This section presents the functional tag set of the treebank. It is divided into four parts, which includes semantic, clause, grammatical and miscellaneous. All four parts have their own labels for annotation depicted in Figure 3.6. Like previous tag sets in Sections 3.3.2.1 and 3.3.2.2, the functional tags are prepared first, then these tags are applied in the manual annotation of the URDU.KON-TB treebank. There are eighteen (18) functional tags in this tag set, which can be added to the syntactic labels by using a dash '-' symbol. A detailed version of SSS tag set after applying functional tags is depicted

(a). *kEs kI samAa2t multavI kar dI jatI hE* 'The hearing of the case is adjourned'



(b). *giraN-faroSoN kE xilAf qAnUn harkat mEN lAyA jAyE* 'The law would be practiced against inflators'

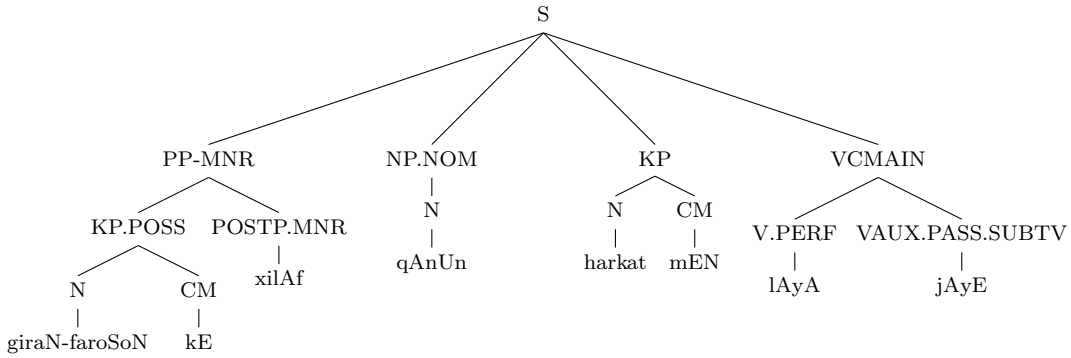


Figure 3.10: A SSS annotation of example KP.POSS, LIGHTV and PP.

in Table 3.7 to understand the hierarchical depth of functional labeling. The labels in semantic part already contributed to SSP and SSS annotation discussed in Sections 3.3.2.1 and 3.3.2.2 respectively. The clause labels like CORR, REL, and RESULT are used to annotate the correlative, relative and resultant clause in a sentence respectively. Grammatical labels include SUB, OBJ, OBJ2, OBL, PLINK and MODF to annotate the subject, object, second object, oblique, predicate link and modifier of a sentence respectively. In the miscellaneous labels, the numbers e.g. 1,2,3, ... are for labeling of antecedents, which are linked to their respective anaphors using a label L e.g. L-1, L-2,

L-3, ... respectively. The empty categories or subcategories are represented with the asterisk '*' symbol. The details of clause and grammatical labels are explained with annotated examples as follows.

Table 3.6: The URDU.KON-TB Functional tagset.

Semantic labels	
CMP (Comparative)	POSS (Possessive)
INST (Instrumental)	SPT (Spatial)
MNR (Manner)	TMP (Temporal)
Clause labels	
CORR (Correlative clause of a sentence)	RESULT (Resultant clause of a sentence)
REL (Relative clause of a sentence)	
Grammatical labels	
MODF (Modifier of a sentence)	OBL (Oblique of a sentence)
OBJ (Object of a sentence)	PLINK (Predicate-Link of a sentence)
OBJ2 (Second Object of a sentence)	SUB (Subject of a sentence)
Miscellaneous labels	
* (for representing empty entities)	L (for linking of anaphors to antecedents)
1,2,3... (for labeling of antecedents)	

The functional annotation of example 3.2 is given in Figure 3.11, whose syntactical annotation was discussed and displayed in Figure 3.7. An ergative case phrase KP.ERG is the subject of the sentence due to presence of an ergative case marker *nE* in the phrase *hAmed nE* 'Hamid' as discussed in the beginning of Section 3.3.2.2. The grammatical label of subject SUB is added to the KP.ERG syntactical label with '-' as KP.ERG-SUB, which represents that this phrase is acting as a subject of the sentence. Another phrase KP.ACC due to accusative case marker *kO* in the phrase *SEr kO* 'the lion' is acting as an object of the sentence and its grammatical label is annotated as KP.ACC-OBJ. The case marker *kO* can be accusative or dative (Butt and King, 2004). Other case phrases in the figure are the KP-SPT (spatial case phrase) and the KP-INST (instrumental case phrase) for *jangal mEN* 'in the jungle' and *bandUq sE* 'with the gun' respectively, which are acting as modifiers or adjuncts in the sentence and annotated with MODF label accordingly. The theories to identify argument structure on the basis of case marking are adopted from Butt and King (2004); Mohanan (1994).

In Figure 3.12(a), the sentence contains a copula verb, which requires two arguments e.g. the subject and the predicate link. The copula verb builds a relation between the subject and the predicate link of the sentence. The noun *te2dAd* 'quantity' without case marker has a nominative case and its syntactic annotation becomes NP.NOM. Since, it

Table 3.7: Detailed hierarchy of functional annotation in the URDU.KON-TB treebank

ADJP (Adjective Phrase)
-MNR (having semantics of Manner)
-PLINK (functioning as Pred. Link of a sentence)
-MODF (acting as a Modifier of a sentence)
-OBL (functioning as Oblique of a sentence)
-PLINK (functioning as Pred. Link of a sentence)
-SPT (having Spatial semantics)
-MODF (functioning as a Modifier)
-TMP (having a Temporal semantics)
-PLINK (acting as a Pred. Link)
ADJPQ (Adjective Phrase contain Question)
-MODF (acting as a Modifier of a sentence)
-OBL (functioning as Oblique of a sentence)
-PLINK (functioning as Pred. Link of a sentence)
ADVP (Adverb Phrase)
-MNR (having semantics of Manner)
-MODF (acting as a Modifier of a sentence)
-MODF (acting as a Modifier of a sentence)
-PLINK (functioning as Pred. Link of a sentence)
-SPT (having Spatial semantics)
-MODF (acting as a Modifier of a sentence)
-OBL (functioning as Oblique of a sentence)
-TMP (having a Temporal semantics)
-MODF (acting as a Modifier of a sentence)
ADVPQ (Adverb Phrase contain Question)
-MNR (having semantics of Manner)
-MODF (acting as a Modifier of a sentence)
-MODF (acting as a Modifier of a sentence)
-TMP (having a Temporal semantics)
-MODF (acting as a Modifier of a sentence)
-OBJ (functioning as an Object of a sentence)
CL (Clause in a sentence)
-MODF (acting as a Modifier of a sentence)
-PLINK (functioning as Pred. Link of a sentence)
CORDP (Coordination Phrase)
.NOM (has a Nominative case)
-OBJ (functioning as an Object of a sentence)
-PLINK (functioning as Pred. Link of a sentence)
-SPT (having Spatial semantics)
-PLINK (functioning as Pred. Link)
-SUB (functioning as a Subject)
-SUB (functioning as Subject of a sentence)
-INST (has an Instrumental case)
-PLINK (functioning as Pred. Link of a sentence)
-MNR (having semantics of Manner)
-MODF (acting as a Modifier of a sentence)
-MODF (acting as a Modifier of a sentence)
-PLINK (functioning as Pred. Link of a sentence)
-SPT (having Spatial semantics)
-TMP (having a Temporal semantics)
DATEP (Date Phrase)
-TMP (having a Temporal semantics)
-MODF (acting as a Modifier of a sentence)
PARP (Parenthetic Phrase)
KP (Case Phrase)

.ACC (has an Accusative case)
-OBJ (functioning as an Object of a sentence)
-SPT (having Spatial semantics)
-OBJ (functioning as an Object)
-SUB (functioning as Subject of a sentence)
.DAT (has a Dative case)
-OBJ (functioning as an Object of a sentence)
-OBJ2 (functioning as a Second Object)
-SPT (having a Spatial semantics)
-OBJ2 (functioning as a Second Object)
-SUB (functioning as a Subject)
-SUB (functioning as Subject of a sentence)
.ERG (has an Ergative case)
-SPT (having Spatial semantics)
-SUB (functioning as a Subject)
-SUB (functioning as Subject of a sentence)
.POSS (has a Possessive case)
-MNR (having semantics of Manner)
-MODF (acting as a Modifier of a sentence)
-OBL (functioning as Oblique of a sentence)
-PLINK (functioning as Pred. Link of a sentence)
-SPT (having a Spatial semantics)
-OBL (functioning as Oblique)
-SUB (functioning as a Subject)
-SUB (functioning as Subject of a sentence)
-TMP (having a Temporal semantics)
-PLINK (functioning as Pred. Link)
-INST (has an Instrumental case)
-MNR (having semantics of Manner)
-MODF (acting as a Modifier)
-OBL (functioning as Oblique)
-MODF (acting as a Modifier of a sentence)
-PLINK (functioning as Pred. Link of a sentence)
-SPT (having a Spatial semantics)
-MODF (acting as a Modifier)
-OBL (functioning as Oblique)
-TMP (having a Temporal semantics)
-MODF (acting as a Modifier)
-OBL (functioning as Oblique)
-MNR (having semantics of Manner)
-MODF (acting as a Modifier of a sentence)
-PLINK (functioning as Pred. Link of a sentence)
-MODF (acting as a Modifier of a sentence)
-OBL (functioning as Oblique of a sentence)
-PLINK (functioning as Pred. Link of a sentence)
-SPT (having a Spatial semantics)
-MODF (acting as a Modifier of a sentence)
-OBL (functioning as Oblique of a sentence)
-PLINK (functioning as Pred. Link of a sentence)
-SUB (functioning as Subject of a sentence)
-TMP (having a Temporal semantics)
-MODF (acting as a Modifier of a sentence)
-OBL (functioning as Oblique of a sentence)
-PLINK (functioning as Pred. Link of a sentence)
KPQ (Case Phrase contain a Question)

.POSS (has a Possessive case)
-INST (has an Instrumental case)
-OBL (functioning as Oblique of a sentence)
NP (Noun Phrase)
.ACC (has an Accusative case)
-OBJ (functioning as an Object of a sentence)
-SUB (functioning as Subject of a sentence)
.DAT (has a Dative case)
-OBJ2 (functioning as a Second Object)
-SUB (functioning as Subject of a sentence)
-MNR (having semantics of Manner)
.NOM (has a Nominative case)
-MNR (having semantics of Manner)
-OBJ (functioning as an Object)
-SUB (functioning as a Subject)
-MODF (acting as a Modifier of a sentence)
-OBJ (functioning as an Object of a sentence)
-OBJ2 (functioning as a Second Object)
-OBL (functioning as an Oblique of a sentence)
-PLINK (functioning as Pred. Link of a sentence)
-SPT (having a Spatial semantics)
-OBJ (functioning as an Object)
-PLINK (functioning as Pred. Link)
-SUB (functioning as a Subject)
-SUB (functioning as Subject of a sentence)
-TMP (having a Temporal semantics)
-OBJ (functioning as an Object)
-OBL (functioning as Oblique)
-PLINK (functioning as Pred. Link)
-SUB (functioning as a Subject)
.POSS (has a Possessive case)
-SUB (functioning as Subject of a sentence)
-MNR (having semantics of Manner)
-MODF (acting as a Modifier of a sentence)
-MODF (acting as a Modifier of a sentence)
-SPT (having a Spatial semantics)
-MODF (acting as a Modifier of a sentence)
-OBJ (functioning as an Object of a sentence)
-OBL (functioning as an Oblique of a sentence)
-TMP (having a Temporal semantics)
-MODF (acting as a Modifier of a sentence)
NPQ (Noun Phrase contain a Question)
.ACC (has an Accusative case)
-OBJ (functioning as an Object of a sentence)
.NOM (has a Nominative case)
-OBJ (functioning as an Object of a sentence)
-PLINK (functioning as Pred. Link of a sentence)
-SUB (functioning as Subject of a sentence)
-OBJ (functioning as an Object of a sentence)
-PLINK (functioning as Pred. Link of a sentence)
-SUB (functioning as Subject of a sentence)
PP (Pre/Post-Position Phrase)
-CMP (has a semantics of Comparison)
-SPT (having a Spatial semantics)
-TMP (having a Temporal semantics)
-INST (has an Instrumental case)
-MNR (having semantics of Manner)
-MODF (acting as a Modifier of a sentence)
-PLINK (functioning as Pred. Link of a sentence)
-MODF (acting as a Modifier of a sentence)
-OBL (functioning as an Oblique of a sentence)
-PLINK (functioning as Pred. Link of a sentence)
-POSS (has a Possessive case)
-SUB (functioning as Subject of a sentence)
-SPT (having a Spatial semantics)
-MODF (acting as a Modifier of a sentence)
-OBL (functioning as an Oblique of a sentence)
-PLINK (functioning as Pred. Link of a sentence)
-TMP (having a Temporal semantics)
-MODF (acting as a Modifier of a sentence)
-OBL (functioning as an Oblique of a sentence)
QP (Quantifier Phrase)
-PLINK (functioning as Pred. Link of a sentence)
-SUB (functioning as Subject of a sentence)
QWP (Question Word Phrase)
S (root of the Sentence structure)
SBAR (Subordinate clause)
-CORR (Correlative clause)
-MODF (acting as a Modifier of a sentence)
-PLINK (functioning as Pred. Link of a sentence)
-REL (Relative clause)
-RESULT (Resultant clause)
SBARQ (Subordinate clause contain Question)
-RESULT (Resultant clause)
SQ (yes/no Question Sentence)
HP (Heading Phrase)
-MODF (acting as a Modifier of a sentence)
UP (Unit Phrase)
VALAP (special Vala Phrase)
-PLINK (functioning as Pred. Link of a sentence)
-SUB (functioning as Subject of a sentence)
VCMAIN (Main Verb Phrase of a sentence)
VCP (Verb Phrase contain Complex Predicate)
-MODF (acting as a Modifier of a sentence)
VIP (Verb Infinitive Phrase)
-MODF (acting as a Modifier of a sentence)
-OBJ (functioning as an Object of a sentence)
-OBL (functioning as an Oblique of a sentence)
-PLINK (functioning as Pred. Link of a sentence)
-SUB (functioning as Subject of a sentence)
VP (Verb Phrase)
-MODF (acting as a Modifier of a sentence)
-PLINK (functioning as Pred. Link of a sentence)

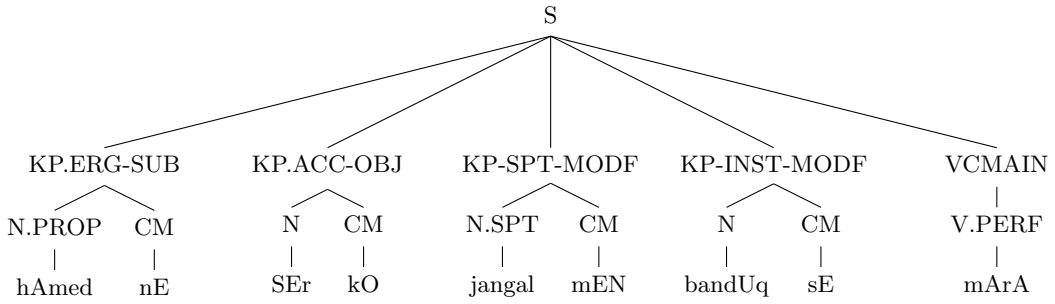
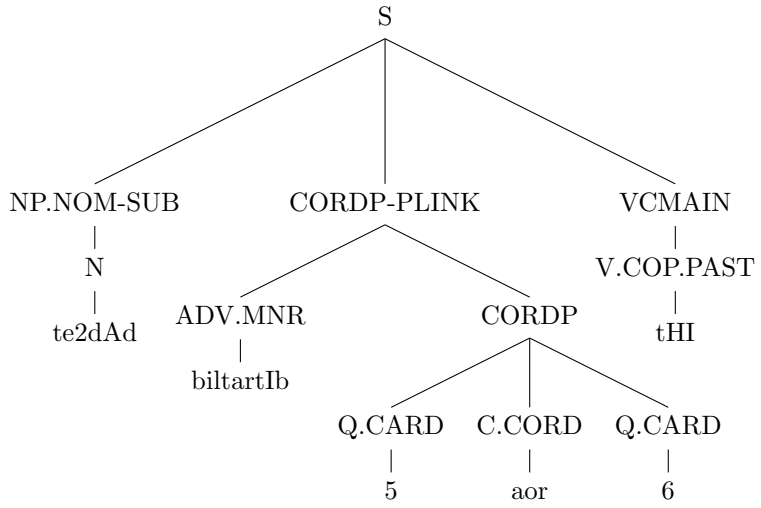


Figure 3.11: Functional annotation of example 3.2

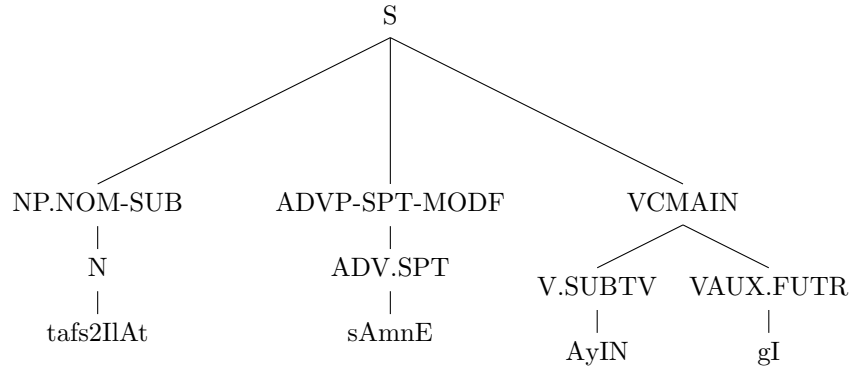
is acting as a subject of the sentence. So, the grammatical label SUB is added to the NP.NOM with a '-' as NP.NOM-SUB. The phrase *biltartIb 5 aor 6* contains a manner-adverb *biltartIb* 'respectively' annotated with ADV.MNR and two cardinal-quantifiers '5' and '6' annotated with Q.CARD with a coordination conjunction *aor* between them, which is then annotated as C.CORD. In our annotation, when a C.CORD appears in a phrase, it is always annotated with CORDP (coordination phrase), which is also acting as a head phrase. As it is functioning as a predicate link PLINK of the sentence, which is then annotated as CORDP-PLINK.

The sentence in Figure 3.12(b) is the simple case of intransitive verb *A* 'come', which essentially needs one grammatical argument called subject. A noun *tafs2IlAt* 'details' is the nominative subject as annotated in the previous example. However, a spatial-adverb *sAmnE* 'front' is functioning as a modifier or an adjunct, which is known as optional elements or elements with secondary importance in the sentence. The tree in Figure 3.12(c) for copula verb has almost the same functional tags as in 3.12(a) except a temporal modifier, a verb infinitive phrase as predicate link and the subject is dislocated due to free order of Urdu. As it is again the copula construction which requires a subject and a predicate link compulsory. The first phrase ADVP-TMP-MODF is acting as a modifier, which is built from a temporal adverb *ab* 'now'. In Urdu, the argument position is not strict due to free order and the next annotated word in the tree is an infinitive verb *dEkHnA* 'to see', which is annotated as VIP (verb infinitive phrase) syntactically. The nominal character or behavior of infinitive verbs in Urdu(Butt, 1995), make it a predicate link annotated with VIP-PLINK in the tree. The final functional argument remains in this sentence is the subject, which is annotated with NP.NOM-SUB for a personal pronoun *yeh* 'it/this' close to VCMAIN in the tree.

(a). *te2dAd biltartIb 5 aor 6 tHI* 'The quantity was 5 and 6 respectively'



(b). *tafs2IlAt sAmnE AyIN gI* 'The details will come out'



(c). *ab dEkHnA yE hE* 'Now, it is to see'

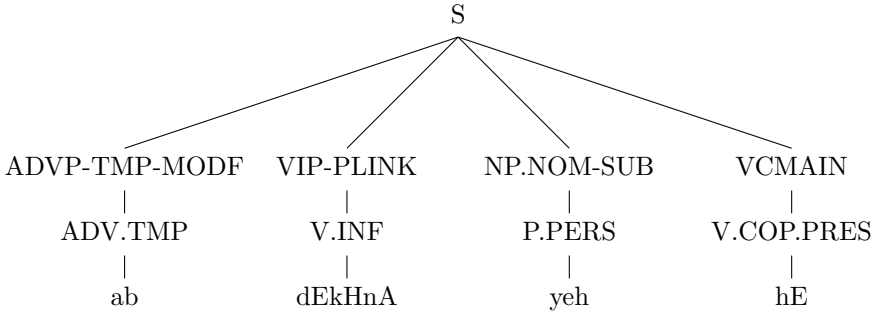


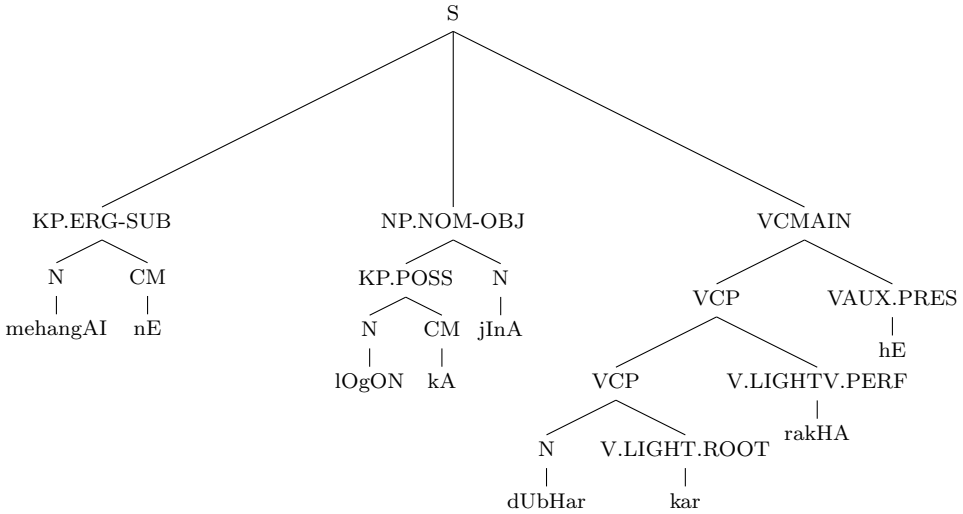
Figure 3.12: A functional annotation of sentences

The functional annotation of sentences containing object and oblique is presented in Figure 3.13. The tree in Figure 3.13(a) contains a transitive complex predicate *dUbHar kar rakHA* 'made hard', which requires an object along with the subject. *dUbHar kar* 'do hard' is making a noun-verb complex predicate and the last verb *kar* 'do' with root form in it is making a verb-verb complex predicate along with the next verb *rakHA* 'kept' as already discussed in Figure 3.10 of Section 3.3.2.2. Similarly, a phrase *mehangAI nE* 'inflation' is the ergative subject of the sentence as discussed in Figure 3.11. The argument, which is not annotated in earlier examples is the object OBJ of this sentence. The phrase *lOgON kA jInA* 'life/living of people' is functioning as a nominative object due to nominal behavior of infinitive verb *jInA* 'living/life' and the requirement of the transitive complex predicate. The existence of an object can also be inferred by considering the sentence without the phrase containing the object. For example, without the phrase containing the object , the remaining part of this sentence would be *mehangAI nE dUbHar kar rakHA hE* 'The inflation made hard'. This sounds incomplete and a question is arisen that the inflation made hard to whom?. The answer of this question is the skipped phrase '*lOgON kA jInA* 'life of people'. This is a hit and trial method to identify the object in a sentence. The object can be nominative, accusative and dative according to previous and current discussion in this thesis.

Similarly, the tree in Figure 3.13(b) contains an intransitive verb with auxiliary as *hUA hE* 'has happened', which does not require an object. The word *izAfah* 'increment' is acting as a nominative subject, but there is some other information *mUqadmAt kI ta2dAd mEN* 'in the number of cases' in the sentence, which can be a modifier or oblique. Modifier is a supplementary information or element in the sentence, without which the sense of the sentence is not disturbed but oblique is an element which is not the object of the sentence but its importance is lying between a modifier and an object. An oblique cannot be accusative or dative. To decide between modifier and oblique, a hit and trial method discussed earlier can be applied. Without the candidate element *mUqadmAt kI ta2dAd mEN* , the remaining part of the sentence would be *izAfah hUA hE* 'the increment has happened'. The sense of the sub sentence is fine but the same question can be arisen as the increment has happened to whom. The answer of the question is the missing part, which cannot be object due to intransitive verb but oblique as annotated with KP-OBL in the tree. The theories about argument identification are adopted from Butt et al. (1999).

To illustrate miscellaneous labels in the functional tag set. A bracketed annotation from the URDU.KON-TB treebank for the example sentence 3.5 is presented as follows. The asterisk '*' is used to represent empty categories or subcategories. The antecedents

(a). *mehangAI nE lOgON kA jInA dUbHar kar rakHA hE* 'The inflation made the life of people hard'



(b). *mUqadmAt kI ta2dAd mEN izAfah hUA hE* 'The increment has happened in the number of cases'

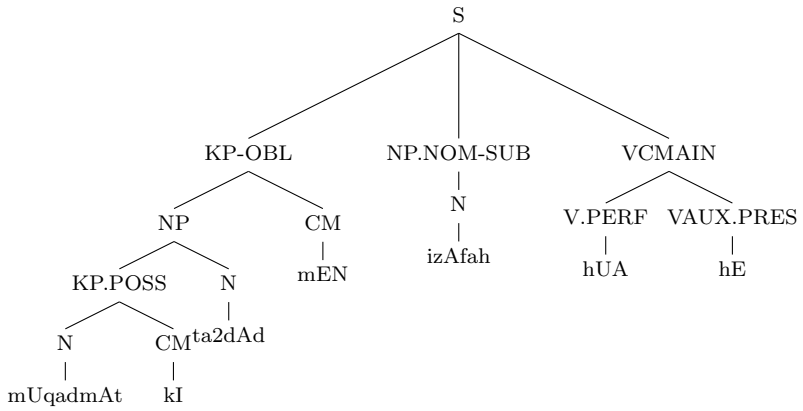


Figure 3.13: A functional annotation of OBJ and OBL

are labelled with numbers e.g. 1 and 2, which are then linked with their anaphors using L as L-1 or L-2 in the bracketed annotation. The example sentence has three parts which include *jis Saxs kE bacE bHUkE pEyAsE hUN* 'the person whose children are hungry and thirsty', *ta2lIm sE meh2rUm hUN* 'are deprived from education' and *usE vardI sE kiyA sarOkAr hE* 'he has no concern with the uniform'. The last part is the subordinating clause and annotated with SBAR. The first two parts are the short version of two sub sentences connected with a coordination conjunction C.CORD, which is absent and annotated with the asterisk '*'. In the second part, who is deprived from education. The answer is the children of the person, which is missing in the beginning of the second part. In treebank development, the annotation of such missing information is helpful for machine text processing tasks. The sentence is annotated with a empty link NP.NOM-SUB-L-1 towards the subject of the first part of the sentence annotated with NP.NOM-SUB-1 for *jis Saxs kE bacE* 'the person, whose children'. In the last part, a personal pronoun *usE* 'he' labeled with L-2 after its POS tag is the anaphoric link of an antecedent *Saxs* 'person' labeled with N-2 in the first part.

(3.5) *jis Saxs kE bacE bHUkE pEyAsE hUN ta2lIm sE meh2rUm hUN usE vardI sE kiyA sarOkAr hE* 'The person whose children are hungry and thirsty, deprive from education, he has no concern with the uniform.'

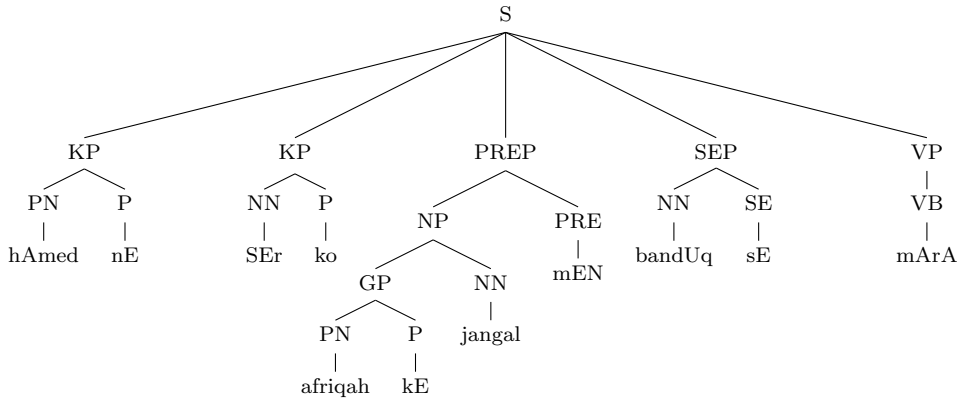
```
( S
   ( NP.NOM-SUB-1
      ( KP.POSS
         ( NP
            ( P.REL.DEM   jis )   ( N-2   Saxs )
         )
         ( CM   kE )
      )
      ( N   bacE )
   )
   ( ADJP-PLINK
      ( ADJ   bHUkE )   ( ADJ   pEyAsE )
   )
   ( VCMAIN
      ( V.COP.SUBTV   hUN )
   )
   ( C.CORD   * )
```

```
        ( NP.NOM-SUB-L-1   * )
        ( KP-INST-OBL
           ( N   ta2lIm )   ( CM   sE )
        )
        ( ADJP-PLINK
           ( ADJ   meh2rUm )
        )
        ( VCMAIN
           ( V.COP.SUBTV   hUN )
        )
        ( SBAR
           ( NP.DAT-SUB
              ( P.PERS-L-2   usE )
           )
           ( KP-INST-OBL
              ( N   vardI )   ( CM   sE )
           )
           ( NPQ.NOM-PLINK
              ( QW   kiyA )   ( N   sarOkAr )
           )
           ( VCMAIN
              ( V.COP.PRES   hE )
           )
           ( M.S   . )
        )
    )
```

3.4 Comparison with the NU-FAST treebank

Section 3.3 completes the discussion of corpus selection and annotation scheme of the URDU.KON-TB treebank. The treebank contains fourteen hundred (1400) annotated sentences in a bracketed form as displayed in example 3.5. An annotated sentence from the NU-FAST treebank (Abbas et al., 2009) is presented in Figure 3.3 for a sentence given in example 3.1, which is repeated in Figure 3.14(a). The figure shows the simplicity of encoded information that contains only limited POS and syntactic tags. There is no functional information in it. The NU-FAST treebank also lacks morpho-

(a). Annotation example of the NU-FAST treebank



(b). Annotation example of the URDU.KON-TB treebank

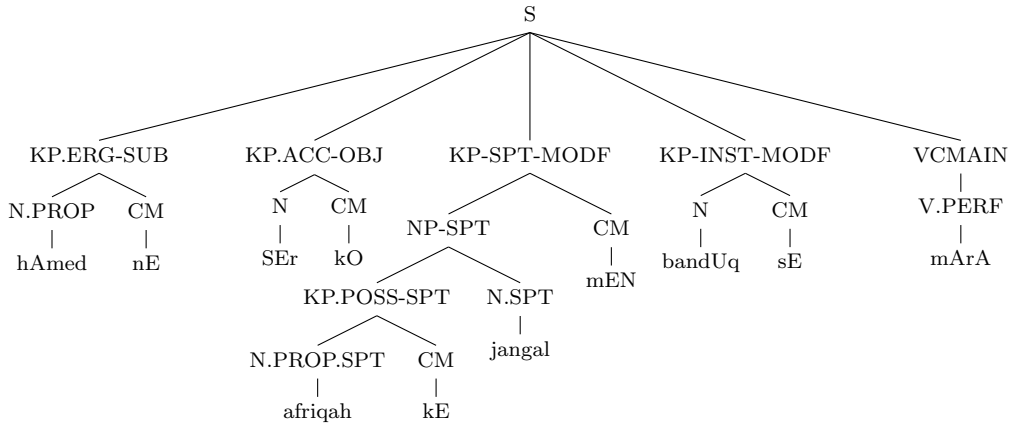


Figure 3.14: Annotation comparison of the NU-FAST and the URDU.KON-TB treebanks

logical information. In contrast to the NU-FAST treebank, the annotation scheme of the URDU.KON-TB treebank has sufficiently rich information. It contains sufficient morphological, POS, syntactical, semantical, clausal, grammatical and miscellaneous information. The same sentence is annotated according to annotation scheme of the URDU.KON-TB treebank, which is presented in Figure 3.14(b) for comparison.

The comparison of Figures 3.14(a) and 3.14(b) shows that the first left most subtree KP in Figure 3.14(a) has no information about ergative case marker *nE* and subject of the sentence. The second left most subtree also lacks information about accusative case marker *kO* and object of the sentence. The third subtree from left does not contain spatial semantics of locations *afriqah* and *jangal*. In Figure 3.14(a), this phrase is

tagged as prepositional phrase, even it is the case of post-positional phrase or a KP. This is due to having only a PRE tag for all types of adpositions. In Figure 3.14(b), the same phrase is annotated as KP-SPT-MODF, which concludes that this phrase has a spatial semantics and acting as a modifier of the sentence. The next subtree in Figure 3.14(b) tells us that the phrase KP-INST-MODF has instrumental meanings and acting as a modifier of the sentence. This information is absent in the same subtree of Figure 3.14(a). The last subtree is about the main verb of the sentence. The verb phrase VP in Figure 3.14(a) is giving information that it is a verb VB and nothing else, while the VCMAIN in Figure 3.14(b) shows that this is the main verb of the sentence having a perfective morphology. In the NU-FAST treebank, if there is more than one verb phrases in a sentence then it has only one tag VP for all of its occurrences and at the end, one cannot infer about the main verb phrase of a sentence. On the other hand, the URDU.KON-TB treebank has a tag VCMAIN for the main verb phrase of the sentence and VP, VIP, VCP for other occurrences in the same sentence. In short, the annotation in the URDU.KON-TB treebank is sufficiently richer than the NU-FAST treebank in respect of morphology, POS, syntactic, semantic, grammatical, etc.

3.5 Summary

In this chapter, the development of an annotated corpus along with its annotation scheme for Urdu has been presented. A combination of PS, DS and HDS has been applied in the annotation of the URDU.KON-TB treebank. A treebank encoded with morphology, POS, syntactic, semantic, clausal, grammatical and miscellaneous information has been concluded. The output of this tree-banking effort proposed a SSP tag set, a SSS tag set, a F tag set and an annotated corpus. These resources can be used for NLP and ML such as probabilistic parsing, training of POS taggers, disambiguation of spoken sentences, grammar development, language identification (Abbas et al., 2010), sources for linguistic inquiry, psychological modeling, pattern matching and in many other applications of this domain (Marcus et al., 1993) & (Bies et al., 1995). After the development of the URDU.KON-TB treebank, an evaluation of the proposed tag sets needed to be performed. This is the topic of the next Chapter 4.

4

Treebank Annotation Evaluation

This chapter describes the evaluation of the Urdu treebank annotation presented in Chapter 3. The evaluation is the process of calculating inter-annotator agreement, which provides a quantitative answer as to the overall consistency plus feasibility of the annotation scheme. For the evaluation of the URDU.KON-TB treebank annotation, the most advanced measure known as the Krippendorff's Alpha α coefficient (Krippendorff, 2004) is used. The output of the annotators is recorded and processed. The reliability of POS, syntactical and functional annotations is evaluated. The issues faced in each type of annotation evaluation are discussed here and the respective revisions which were made are also reported in this chapter.

4.1 Introduction

Treebank evaluation is the process of evaluating the reliability (consistency plus feasibility) of a treebank annotation. This can be measured by calculating the agreement among annotators of a treebank. The reliability evaluation is a complex task for a treebank that contains rich information but is an essential part in this production of a good quality treebank. The process of evaluation has recently seen some advancements. Evaluators can calculate the inter-annotator agreement (IAA) along with the annotator's individual error rate. The error rate is then used to calculate the performance and efficiency of each individual annotator (Mikulová and Stepánek, 2010).

Some generally accepted statistical methods exist for the evaluation of annotated data. These methods include S (Bennett et al., 1954), π (Scott, 1955), κ (Cohen et al., 1960) and α (Krippendorff, 2004, 2012) used for either single, double or multi (more than two) annotators/coders. As per recommendations of Artstein and Poesio (2008),

the minimum number of annotators required is three, but more is appropriate. For the annotation evaluation of the URDU.KON-TB treebank, a class of twenty five (25) students was trained. The annotation work was assigned to them as homework (see Section 4.3.1). The annotators were not provided by choices during the annotation of data, which means the nominal data (tags assigned by the annotators) is not fixed but random. More than two annotators and the random nominal data were the two requirements of our annotation evaluation, for which I investigated the existing evaluation measures presented as follows.

4.1.1 Overview & Choices in Methods

The statistical method S for the evaluation of consistency is used for a single annotator, which is further modified for two annotators by Bennett (1972). It does not meet our requirement of more than two annotators and random nominal data. So, it is dropped from the list of candidates for annotation evaluation without any further discussion. In the beginning, π and κ were not for multi-annotators but later on, these were modified for multi-annotators and named as multi-π and multi-κ respectively. These remaining three methods π, κ and α in our candidate list are commonly used to measure the agreement/disagreement among the annotators. At this point, it is to be decided which one is the best among these three methods of annotation evaluation.

For the annotation evaluation of the URDU.KON-TB treebank, the Krippendorff's α co-efficient was ultimately selected due to the following reasons.

1. Both multi-π and multi-κ have a serious limitation in that all disagreements are treated equally in contrast to α (Artstein and Poesio, 2008; Bruce et al., 1998; Carletta et al., 1997; Carlson et al., 2003; Core and Allen, 1997; Craggs and Wood, 2004; Di Eugenio, 2000; Marcu et al., 1999; Mieskes and Strube, 2006; Poesio and Vieira, 1998; Stevenson and Gaizauskas, 2000; Stolcke et al., 2000; Véronis, 1998). This is because multi-π and multi-κ only considered agreements of annotators.

2. Multi-κ has no solution to resolve bias except that the number of annotators should be increased (Artstein and Poesio, 2008; Brennan and Prediger, 1981; Craggs and Wood, 2005; Di Eugenio and Glass, 2004; Siegel and Castellan, 1988; Zwick, 1988).

3. Multi-π and multi-κ do not address the issue of prevalence which means annotations with few common categories gives you high accuracy (Artstein and Poesio, 2008; Cicchetti and Feinstein, 1990; Di Eugenio and Glass, 2004; Feinstein and Cicchetti, 1990).

4. π and κ were designed for two annotators and their modified versions are not fully equipped for use of more than two annotators (Artstein and Poesio, 2008; Carletta et al., 1997).

5. Both multi-π and multi-κ are designed to handle fixed nominal data.

Krippendorff's α uses the same assumptions as π and κ but evaluates a different magnitude of disagreement and missing values (Krippendorff, 2004). The problems 2 and 3 related to bias and prevalence are minor issues and can be controlled by increasing the number of annotators and a tag set with sufficient tags (not few) respectively. The problems of multi-π and multi-κ mentioned in 1 and 4 are critical and are discussed in Sections 4.2.2.1 and 4.2.2.2, respectively. If the tag set is hierarchical as in our case, it is recommended to use Krippendorff's α co-efficient for annotation evaluation (Di Eugenio and Glass, 2004; Passonneau, 2004). The hierarchical tag sets of the URDU.KON-TB treebank contain many number of tags and an annotator can assign any tag to any token of a sentence without having particular choices. This means that the nominal data is not fixed, so the choice of multi-π and multi-κ is not suitable. The annotation evaluation results of the URDU.KON-TB treebank using α are presented in Sections 4.3.2, 4.3.3 and 4.3.4 for POS, syntactical and functional annotation, respectively. The work done before evaluation is mentioned in Section 4.3.1, while the revision of tags after evaluation for each annotation type is presented in Sections 4.4.1, 4.4.2 and 4.4.3, respectively.

Initially, the URDU.KON-TB tree bank was annotated according to the guidelines devised by Abbas (2012). Before the final annotation evaluation, a successive process of minor revisions was undertaken and the annotation guidelines were updated accordingly. The process of annotation evaluation was performed by using these revised guidelines, which were revised again after the evaluation process. The most updated annotation guidelines are presented in Chapter 2. The reliability of the POS, syntactical and functional annotation of the URDU.KON-TB treebank obtained is 0.964, 0.817 and 0.806, respectively. This is presented in Sections 4.3.2, 4.3.3 and 4.3.4, respectively.

4.2 Quality Control Methods for Corpus Annotations

4.2.1 Previous Work

In this section, some recent work is reported, in which most of the researchers used the methods for measuring the IAA described in Section 4.1. This work includes PropBank (Palmer et al., 2007), OntoNotes (Hovy et al., 2006), Urdu Dependency Treebank

(UDT) (Bhat and Sharma, 2012a), corpus for metonymies (Markert and Nissim, 2002), NEGRA corpus (Brants, 2000a) and the SALSA corpus (Erk et al., 2003). In the work of PropBank (Palmer et al., 2005, 2007), two annotators were used to evaluate the annotation of PropBank. They explicitly mention that the process of evaluating the inter-annotator agreement for the PropBank is complicated because each verb has a large number of possible annotations (Palmer et al., 2005, p. 86), which means that the nominal data (annotation tags) were not fixed. They used κ by Siegel and Castellan Jr (1988), which is a fixed marginal κ and can be used in a situation when the annotators are forced to assign certain number of annotation tags to categories. It would have been better, if they had used the free marginal κ (Randolph et al., 2005; Warrens, 2010) or the best available measure α by Krippendorff (2004) in this situation. In their work, the κ value achieved for the combined identification and classification decision is 0.91 including all sub types of arguments, which means their annotation is reliable.

The main aim of OntoNotes was the construction of a multilingual annotated corpus for English, Chinese and Arabic. The resources built during the development of the PropBank were reused during the development of the OntoNotes. They reported an inter-annotator agreement of 90% using the same κ as was practiced in the work of the PropBank. So, the same observations are applied on the calculation of their IAA.

In the work of UDT for Urdu, the measure proposed by Cohen et al. (1960) was used. For interpretation of κ values, they used the scale proposed in Landis and Koch (1977). Two annotators were used to annotate 196 sentences. Their respective agreements and disagreements were recorded and the κ value was computed as 0.71, which lies at the level of substantial agreement according to Landis and Koch (1977). The scale proposed by Landis and Koch (1977) is considered to be a very weak scale in the field of computational linguistics (CL) as narrated in Carletta (1996); Gwet (2012); Krippendorff (2004, 2012); Neuendorf (2002). The number of annotators used and the choice of using Cohen's κ is criticized in the recent literature and is discussed in Section 4.2.2.2. Similarly, in another work of corpus construction for metonymies (the case of location names) by Markert and Nissim (2002), the same measure of Cohen's κ is adopted but details of the statistics are not provided. They reported IAA ranging from 0.80 to 0.88 for different categories. The interpretation of κ values is performed according to α measure discussed in Krippendorff (2012).

The NEGRA corpus which is a collection of German newspaper text used a different method for evaluating the IAA. They used the PARSEVAL (Black et al., 1991) measure for evaluation of IAA. Two expert annotators were provided a graphical user interface (GUI) (Brants, 2000b; Brants and Plaehn, 2000) for annotation. The anno-

tators performed the syntactic annotation with POS and structure information. The IAA for POS is reported as 98.6%, while the labeled F-score reported for structural annotation is 92.4%. The accuracy is computed on the basis of identical annotated tags between two annotators. The values which are not identical are treated equally like multi-π and multi-κ discussed in Sections 4.2.2.1 and 4.2.2.2. The measure has no solution for annotators bias and prevalence problem. Moreover, if the annotated data has missing values then these would not be counted.

The SALSA (The Saarbrücken Lexical Semantics Acquisition) project used already developed resources in FrameNet project (Johnson et al., 2002) and the TIGER treebank project (Brants et al., 2002) for a construction of a large semantically annotated corpus. The annotation is evaluated through Krippendorff's α coefficient. For their annotation task, they used two annotators and the data given to the annotators was not random. In all of his articles, Krippendorff advocated to use more than two annotators and a minimum number of annotators recommended is three. For getting more reliable and quality results, it is advised to use random data for annotation. The α values reported for IAA in SALSA project are 85% on frames and 86% on roles.

4.2.2 Measuring Inter-Annotator Agreement (IAA)

To understand the use and difference among multi-π, multi-κ and α, the content here with respective examples is divided into proper subsections as follows. However, Sections 4.2.2.1 and 4.2.2.2 can be skipped if one already has a knowledge of the controversies and issues involved in them.

4.2.2.1 Multi-Pi

This method is basically the generalization of Scott's π modified by Fleiss (1971). He used the same kappa κ convention in his work, which created a misunderstanding that either this proposed method is the generalization of Scott's π (Scott, 1955) or Cohen's κ (Cohen et al., 1960). The general form of multi-π is given in Equation 4.1.

$$\kappa = \frac{\bar{A} - \bar{A}_e}{1 - \bar{A}_e} \tag{4.1}$$

The numerator of the equation measures the actual amount of agreement beyond chance and denominator measures the attainable amount of agreement above chance. The ratio between numerator and denominator measures the proportion of actual observed possible agreement beyond chance. For complete agreement between annotators, the value of κ becomes 1 and for no agreement, it is less than or equal to 0. The $\bar{A}$ is

the mean of A_i's as can be seen in Equation 4.2.

$$\bar{A} = \frac{1}{T}\sum_{i=1}^{T} A_i = \frac{1}{Tn(n-1)}(\sum_{i=1}^{T}\sum_{j=1}^{t} n_{ij}^2 - Tn) \tag{4.2}$$

To evaluate $\bar{A}$, we must know different variables used in this equation. Here T is the total number of candidates tokens for annotation, t is the number of tags to be annotated and n is the number of annotators agreed to assign particular tag to a particular candidate token. From Equation 4.2, it can be seen that A_i should be computed first. A_i measures the annotators agreement for the i-th token. The formula used is depicted in Equation 4.3. The value of i varies from 1 to T and the value of j varies from 1 to m. Similarly, n_ij is the number of annotators assigned the j-th tag to i-th token.

$$A_i = \frac{1}{n(n-1)}\sum_{j=1}^{m} n_{ij}(n_{ij}-1) = \frac{1}{n(n-1)}\sum_{j=1}^{m}(n_{ij}^2 - n_{ij}) = \frac{1}{n(n-1)}[(\sum_{j=1}^{m} n_{ij}^2) - (n)] \tag{4.3}$$

At this point, the half of the portion of Equation 4.1 is computed. The other half is dependent on computation of another variable $\bar{A}_e$, which can be seen in Equation 4.4.

$$\bar{A}_e = \sum_{j=1}^{t} a_j^2 \tag{4.4}$$

Here, a_j should be computed before the evaluation of $\bar{A}_e$. The formula of a_j is given in Equation 4.5, which is the ratio of all i-th assigned tokens with the j-th tag.

$$a_j = \frac{1}{Tn}\sum_{i=1}^{T} n_{ij} \tag{4.5}$$

To evaluate the value of multi-π according to POS annotation of the URDU.KON-TB treebank, a sentence *yeh kitAb hE* is given as follows. The sentence is encoded with actual POS annotation, however, the annotators can assign a different tag from the available POS annotation tags.

yeh	*kitAb*	*hE.*
P.PERS	N	V.COP.PRES

'This is a book'

This sentence contains T=3 number of candidate tokens and suppose t=5 different tags assigned to respective tokens by n=5 annotators. The annotated data by annotators is first recorded in a tabular form as shown in Table 4.1. The cell values represent the

Table 4.1: Multi- πomputation

Tokens \ Tags	P.PERS	P.DEM	N	V.COP.PRES	VAUX.PRES	A_i
yeh	4	1	0	0	0	0.600
kitAb	0	0	5	0	0	1.000
hE	0	0	0	4	1	0.600
Total	4	1	5	4	1	
a_j	0.267	0.067	0.333	0.267	0.067	

number of annotators assign the respective tag in the column to respective token in the row of the table. To illustrate the computation of a_j in the table, the value of a_1 for the column P.PERS is calculated according to Equation 4.5 as $a_1 = \frac{1}{3*5}\sum_{i=1}^{3} n_{i1} = \frac{4+0+0}{15} = 0.267$. The remaining values of a_j are calculated in a same way and recorded in the Table 4.1. By putting the individual results of Equation 4.5 into Equation 4.4, we get the final result for $\bar{A}_e$ in Equation 4.6 as follows.

$$\bar{A}_e = \sum_{j=1}^{t} a_j^2 = 0.267^2 + 0.067^2 + 0.333^2 + 0.267^2 + 0.067^2 = 0.262 \tag{4.6}$$

Now, we have to calculate the values for Equation 4.3. The value of A_1 for token *yeh* can be calculated as $A_1 = \frac{1}{5(5-1)}[(\sum_{j=1}^{5} n_{1j}^2)-(5)] = \frac{1}{5(5-1)}[(4^2+1^2+0^2+0^2+0^2)-(5)] = 0.600$. Similarly, the other values of column A_i in the table are calculated and recorded in the table according to Equation 4.3. As this equation is the part of Equation 4.2, so by putting the resultant values of Equation 4.3 into Equation 4.2, we obtained the resultant value for $\bar{A}$ in Equation 4.7 as follows.

$$\bar{A} = \frac{1}{T}\sum_{i=1}^{T} A_i = \frac{1}{3}[(0.600 + 1.000 + 0.600)] = 0.733 \tag{4.7}$$

Now, by placing the resultant values of Equations 4.6 and 4.7 into Equation 4.1, we get the final value for multi-π as $\kappa = \frac{\bar{A}-\bar{A}_e}{1-\bar{A}_e} = \frac{0.733-0.262}{1-0.262} = 0.639$. At this stage, the value obtained for κ has to be compared with the range of a common scale available for S, π, κ and α statistical methods. This scale was proposed by Landis and Koch (1977) and is given in Figure 4.1. This scale is not authentic because Landis and Koch (1977) did not provide any evidence to support it (Gwet, 2012). This became the reason that Fleiss et al. (2013) gave their own version of the scale. According to Fleiss, a κ value over 0.75 is excellent, a value between 0.40 to 0.75 varies from fair to good and a value less than 0.40 is considered to be a poor reliability.

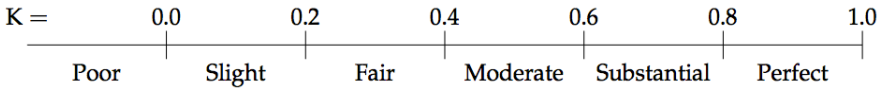


Figure 4.1: Agreement values and strength by Landis and Koch (1977)

4.2.2.2 Multi-Kappa

Multi-κ is the generalization of Cohen's κ devised by various authors e.g. Light (1971), Fleiss (1971), Hubert (1977) and Conger (1980). The generalization of Cohen's κ proposed by Fleiss (1971) is the generalization of Scott's π according to Artstein and Poesio (2008); Craig (1981); Hayes and Krippendorff (2007); Warrens (2010), etc. presented earlier in Section 4.2.2.1. The statistical measures for evaluation of agreement by Hubert (1977) and Conger (1980) are the same but are proposed independently as claimed in Davies and Fleiss (1982); Heuvelmans and Sanders (1993); Popping (1983). The measures by Light (1971) and Hubert (1977) are the true generalization of Cohen's κ. These modified versions are based on Cohen's κ. Due to limitations of Cohen's κ and its generalized versions by Light (1971) and Hubert (1977), the use of multi-κ is not recommended for the evaluation of an annotation's reliability (Craggs and Wood, 2005; Krippendorff, 2004, 2012). Cohen's κ was devised for two annotators but can be used for more than two annotators (Sim and Wright, 2005) with limitations (Artstein and Poesio, 2008). The general equation used for Cohen's κ is similar to Equation 4.1, which is revised in Equation 4.8 with different variable names according to Cohen's work (Cohen et al., 1960) as follows.

$$\kappa = \frac{A_o - A_c}{1 - A_c} \tag{4.8}$$

Here A_o is observed agreement and A_c is the agreement by chance. The definition of Equation 4.8 is same as of Equation 4.1, which states the proportion of agreement beyond chance.

To illustrate κ calculation, the sentence example given in Section 4.2.2.1 is hereby taken with all suppositions described there. We assume five annotators, labelled as A,B,C,D and E, and the sentence contains $T = 3$ tokens. In this method, κ values for each pair of annotators are calculated first and then their average is evaluated. Table 4.2 represents the data of a pair (A,B) of annotators, in which both the annotators provide 1/3 correct tags without any further information. The observed percentage of agreement A_o can be calculated by summing up the diagonal values and then divide

Table 4.2: Multi-κomputation

A \ B	Correct	Incorrect	Sum	Probability
Correct	1	0	1	0.33
Incorrect	0	0	0	0.0
Sum	1	0	1	
Probability	0.33	0.0		

the sum by T. Thus, it evaluates to $A_o = 1 + 0/3 = 0.33$. By summing up the rows we obtained 1 and 0 correct and incorrect annotations respectively by A and similarly, by summing up the columns we obtained 1 and 0 correct and incorrect annotations respectively by B. The random probability that both of the annotators (A and B) would provide correct tags is calculated by multiplying the respective values as $0.33 * 0.33 = 0.11$ and similarly for incorrect tags as $0.0*0.0. = 0.0$. The overall probability of random agreement is obtained by summing up the respective values as $A_c = 0.11 + 0.0 = 0.11$. By the putting the resultant values of A_o and A_c in Equation 4.8, the κ value obtained for a pair (A,B) is presented in Equation 4.9 as follows.

$$\kappa_{A,B} = \frac{0.33 - 0.11}{1 - 0.11} = 0.25 \tag{4.9}$$

The κ values for other pairs $(\kappa_{A,C}, \kappa_{A,D}, \kappa_{A,E}, \kappa_{B,C}, \kappa_{B,D}, \kappa_{B,E}, \kappa_{C,D}, \kappa_{C,E}, \kappa_{D,E})$ can be calculated in the same way as depicted in Equation 4.9. The average Cohen's κ of all pairs can be obtained by using Equation 4.10 as follows.

$$\kappa = \frac{\kappa_{A,B} + \kappa_{A,C} + \kappa_{A,D} + \kappa_{A,E} + \kappa_{B,C} + \kappa_{B,D} + \kappa_{B,E} + \kappa_{C,D} + \kappa_{C,E} + \kappa_{D,E}}{10} \tag{4.10}$$

$$= \frac{0.25 + 0.40 + 1.00 + 0.66 + 0.78 + 0.88 + 0.9 + 0.12 + 0.56 + 0.25}{10}$$

$$= 0.58$$

Cohen adopted Pearson's correlation coefficient (Pearson et al., 1901) ranging from -1 to +1. The values +1, -1 and 0 indicate perfect agreement, perfect disagreement and no relationship between two annotators respectively. The value 0.70 is considered to be a satisfactory agreement. The researchers are also using a scale for evaluation of their κ values devised by Landis and Koch (1977), which is presented in previous Section

Table 4.3: Karl Pearson Correlation Coefficient r

	Correlation r	
Association	Positive	Negative
Small	0.1 to 0.3	-0.1 to -0.3
Medium	0.3 to 0.5	-0.3 to -0.5
Large	0.5 to 1.0	-0.5 to -1.0

4.2.2.1. The scale devised by Karl Pearson is given in Table 4.3 for further detailed ranges.

To calculate overall percentage agreement, consider the same example sentence of Section 4.2.2.1 with three annotators. Table 4.4 shows the annotation recording of three annotators (A, B and C). The columns show the labeling of annotators and the rows depict the gold standard (GS) tags for tokens of the example sentence. The cell values of the table are tags awarded by the annotators. With the help of Table 4.4, the

Table 4.4: Overall Percentage Agreement for Multi Annotators

GS Tags	A	B	C
P.PERS	P.PERS	P.PERS	P.DEM
N	N	N	N
V.COP.PRES	VAUX.PRES	V.COP.PRES	VAUX.PRES

counts of pairwise agreement are calculated and depicted in Table 4.5. For example the annotators A and B have a perfect agreement in assigning the same tag P.PERS in Table 4.4 and recorded as 1 in the first cell of Table 4.5. The next step is to calculate the sum of each column of Table 4.5. The total GS tags for the example sentence is 3, which are mentioned in a row of total ratings. The proportion of pairwise agreement is calculated after division of respective sum and total rating values. The proportions are then converted into respective percentages. The mean of the pairwise percentage agreements is thus 56% as presented in Table 4.5. The simplicity of this measure is the reason that most of the linguists adopted this method to evaluate their annotation schemes as exercised in Propbank (Palmer et al., 2007), OntoNotes annotation (Hovy et al., 2006), Urdu dependency treebank (Bhat and Sharma, 2012a), etc.

Table 4.5: Counts of Pairwise Agreement

	A-B	A-C	B-C
	1	0	0
	1	1	1
	0	1	0
Sum	2	2	1
Total Ratings	3	3	3
Proportion Agreement	0.67	0.67	0.33
Percentage Agreement	67%	67%	33%
Overall Percentage Agreement	56%		

4.2.2.3 Krippendorff's Alpha

Krippendorff's α uses the same assumptions as π and κ but evaluates a different magnitude of disagreement and missing values (Krippendorff, 2004). It enables researchers to judge a variety of data with same reliability standard. In contrast to multi-π and multi-κ, it allows any number of annotators. The minimum number of annotators allowed is three but more than three is appropriate according to Krippendorff's recommendations. It has different levels of measurement to measure binary, nominal, ordinal, interval, ratio, etc. type of data. Krippendorff's α (Krippendorff, 1970, 2004) satisfies all the issues existed in multi-π and multi-κ discussed in Sections 4.2.2.1 and 4.2.2.2 and has become the standard reliability measure for data analysis (Hayes and Krippendorff, 2007). Any number of categories, scale values and levels of measurement can be used. The categories are the items to which the annotators assigned the values e.g. tokens are assigned tags by annotators in Section 4.2.2.1. It can handle empty values and reliability of data is not affected by large or small size of annotated data. It is claimed as a most advanced and reviewed measure and no other measure is able to compete with it (Hayes and Krippendorff, 2007).

The general formula used to compute α reliability by Krippendorff (1970, 2004) is given in Equation 4.11, where O_d is the observed disagreement and E_d is the expected disagreement. For perfect agreement among annotators, α becomes 1 and $\alpha = 0$ is for absence of agreement. The respective forms of O_d and E_d are presented in Equations 4.12 and 4.13 respectively. The following variables o_{pq}, n_p, n_q and n are the frequencies

of values in coincidence matrix, which will be constructed later.

$$\alpha = 1 - \frac{O_d}{E_d} \tag{4.11}$$

$$O_d = \frac{1}{n}\sum_p \sum_q o_{pq}\ {}_{metric}\delta^2_{pq} \tag{4.12}$$

$$E_d = \frac{1}{n(n-1)}\sum_p \sum_q n_p.n_q\ {}_{metric}\delta^2_{pq} \tag{4.13}$$

Generally, it evaluates four processes which includes (1) Construction of ***reliability data matrix***, (2) Tabulation of ***coincidence matrix***, (3) Insertion of ***difference function*** ${}_{metric}\delta^2_{pq}$ and finally (4) Computation of α ***reliability***. There are different methods with respect to data type e.g. nominal, ordinal, ratio, interval, etc. for two or more than two annotators. The annotated data type of the URDU.KON-TB treebank is nominal but assigning of tags to tokens by annotators is not limited as normally exist in any fixed scale study e.g. 3 category scale, 5 category scale, etc. In this situation, the general form of Krippendorff's α coefficient is selected according to our requirement, which is explained as follows by taking an example sentence given in Section 4.2.2.1.

Let $x = 5$ be the number of annotators for $N = 3$ number of tokens (units) in an example sentence. After explicit annotation by annotators, the data is recorded according to the reliability data matrix of x by N values given in Table 4.6. $A1$ to $A5$ are the annotators, who assigned different tags e.g. P.PERS, P.DEM, N, V.COP.PRES and VAUX.PRES to tokens (yeh, kitAb and hE) of the example sentence taken in previous Sections 4.2.2.1 and 4.2.2.2. Tokens (units) are represented in columns and annotators are represented in rows of Table 4.6. If some values are missing then the reliability data matrix contains fewer values than xN. The lone value in a unit column can not be paired and ignored during construction of coincidence matrix.

Table 4.6: Reliability Data Matrix

Annotators \ Tokens	yeh	kitAb	hE
$A1$	P.PERS	N	V.COP.PRES
$A2$	P.PERS	N	V.COP.PRES
$A3$	P.DEM	N	V.COP.PRES
$A4$	P.PERS	N	VAUX.PRES
$A5$	P.PERS	N	V.COP.PRES

After the construction of reliability data matrix, if the general method of Krippendorff's α is being in use and the data is nominal then a values (annotated values) by units matrix is to be calculated instead of step 2 and 3 mentioned in this section earlier. After enumerating the annotated values (A-values) of reliability data matrix, the values by units (tokens) matrix obtained is shown in Table 4.7. In this matrix, four annotators assigned the tag P.PERS (personal pronoun) and one annotator assigned the tag P.DEM (demonstrative pronoun) to token *yeh* of the sentence given in the first unit *yeh* column of Table 4.7. The other columns can be narrated in the same way.

Table 4.7: A values by units (tokens) matrix

A-values \ Tokens	yeh	kitAb	hE	Sum	$n_{.1} * (n_{.2} : n_{.5})$	$n_{.2} * (n_{.3} : n_{.5})$	$n_{.3} * (n_{.4} : n_{.5})$	$n_{.4} * n_{.5}$
P.PERS	4	0	0	4 ($n_{.1}$)				
N	0	5	0	5 ($n_{.2}$)	20			
V.COP.PRES	0	0	4	4 ($n_{.3}$)	16	20		
VAUX.PRES	0	0	1	1 ($n_{.4}$)	4	5	4	
P.DEM	1	0	0	1 ($n_{.5}$)	4	5	4	1
Sum	$5(n_{1.})$	$5(n_{2.})$	$5(n_{3.})$	**15**($n..$)	44	30	8	1
$n.. - 1$	14							
$n_{tp}n_{tq}$	4	0	4					
$n_{t.} - 1$	4	4	4					
$\frac{n_{tp}n_{tq}}{n_{t.}-1}$	1	0	1					
$\sum_t \frac{1}{n_{t.}-1} \sum_p \sum_{q>p} n_{tp}n_{tq}\ {}_{metric}\delta^2_{pq}$	2							
$\sum_p \sum_{q>p} n_{.p}n_{.q}\ {}_{metric}\delta^2_{pq}$	83							

As a last step, we have to compute α with an appropriate metric difference function. By using Equations 4.11, 4.12 and 4.13, the final aggregate Equation 4.14 takes the following form. The sum of each token column and A-value row is counted and stored respectively under the heading of *Sum* relatively. According to formula, $n.. = 15$ is the resultant sum highlighted in the Table 4.7, which should be equal to the sum of respective column values ($n_{.1}$ to $n_{.5}$) and row values ($n_{1.}$ to $n_{3.}$). Each value of column $(n_{.1} * (n_{.2} : n_{.5}))$ is the by product of $n_{.1}$ with each value of column array $(n_{.2} : n_{.5})$. The values for other columns are calculated in the same way.

$$ {}_{metric}\alpha = 1 - \frac{O_d}{E_d} = 1 - (n.. - 1)\frac{\sum_t \frac{1}{n_{t.}-1} \sum_p \sum_{q>p} n_{tp}n_{tq}\ {}_{metric}\delta^2_{pq}}{\sum_p \sum_{q>p} n_{.p}n_{.q}\ {}_{metric}\delta^2_{pq}} \tag{4.14} $$

According to Equation 4.14 and Table 4.7, the value of $n.. - 1$ is 14 and the product $n_{tp}n_{tq}$ of each token t pairs is 4, 0, and 4 respectively. The values in row $n_{t.} - 1$ are obtained by subtracting 1 from $n_{1.}$, $n_{2.}$ and $n_{3.}$ shown in Table 4.7. After dividing

the values according to a fraction $\frac{n_{tp}n_{tq}}{n_{t.}-1}$, which is the part of numerator of Equation 4.14, the respective resultant values obtained are 1, 0 and 1. Similarly, the result of the whole numerator $\sum_t \frac{1}{n_{t.}-1} \sum_p \sum_{q>p} n_{tp}n_{tq}\ {}_{metric}\delta^2_{pq}$ of equation is evaluated as 2 depicted in the last row of Table 4.7. For nominal data, the perfect agreement is seen as 0 disagreement, so the difference function ${}_{metric}\delta^2_{pq}$ =0. The sum of each column $n_{.1} * (n_{.2} : n_{.5})$, $n_{.2} * (n_{.3} : n_{.5})$, $n_{.3} * (n_{.4} : n_{.5})$ and $n_{.4} * n_{.5}$ is calculated and displayed as 44, 30, 8 and 1 respectively. This sum of each column contributes to evaluate the denominator $\sum_p \sum_{q>p} n_{.p}n_{.q}\ {}_{metric}\delta^2_{pq}$ of Equation 4.14 as 83 after adding up all the sum's values. At the end, by putting all the calculated values in Equation 4.14, it becomes as ${}_{metric}\alpha = 1 - \frac{O_d}{E_d} = 1 - (14)\frac{2}{83} = 0.67$. According to Krippendorff (2004), the value of $\alpha >= 0.80$ is a good reliability. The value of $\alpha < 0.80$ and $\alpha >= 0.67$ states that the tentative conclusions are possible, while $\alpha < 0.67$ describes that a great disagreement is existed among the annotators.

4.2.3 Summary

Coefficients to measure agreement like Cohen's κ and Krippendorff's α have been used by linguists since Carletta (1996) and Passonneau and Litman (1996). However, these measures became really popular after the publication of Di Eugenio and Glass (2004) and Passonneau (2004). The problems of bias and prevalence are the minor issue and can be controlled by increasing the number of annotators and a tag set with sufficient tags (not few) respectively. In our annotation, we used five annotators and the tag sets are hierarchical with many tags having rich information. These rich tag sets were introduced in Chapter 3. Measures like Cohen's κ are not suitable for measuring agreement and many questions have been raised in the studies like Carletta et al. (1997); Carlson et al. (2003); Craggs and Wood (2005); Di Eugenio (2000); Poesio and Vieira (1998); Stolcke et al. (2000) and Core and Allen (1997). These questions include issues of efficiency, whether one can handle more than two coders or not, weak acceptable level of agreement, bias, prevalence, etc. On the one hand, there is sufficient criticism against Cohen's κ and on the other hand, there is a measure like Krippendorff's α, which fulfills our requirement for hierarchical tag sets and five number of annotators. Hence, the Krippendorff's α becomes our choice for annotation evaluation.

4.3 Evaluation

4.3.1 Setup

For the reliability evaluation of annotation guidelines prepared for the URDU.KON-TB treebank of Urdu, it was essential that annotators should be the native speakers of Urdu possessing linguistics skills. To fulfill this purpose, a undergraduate class of 25 linguistics students has been adopted in the training course of annotation at Department of English, University of Sargodha, Pakistan. This training was given to students as a partial part of their major course of linguistics. During this training course, thirty two (32) lectures on annotation guidelines with practical sessions were delivered. The duration of each lecture was of 3 hours. The class was further divided into five groups and during their initial practical sessions, one student with high caliber of understanding was selected (but not informed) secretly from each group for the final annotation. The annotation task of 100 random sentences was divided into 10 home assignments. Each assignment contained 10 sentences. After twenty days of this course, the annotation assignments were given to all students along with the selected students with an instruction not to discuss it with each other. These assignments were collected, marked and the students were awarded with grades. The annotation performed in their home assignments by the selected students was then recorded in Microsoft Excel and evaluated for the reliability of annotation or IAA by applying the Krippendorff's α coefficient. The details of the POS, syntactic and the functional annotation evaluation can be seen in Sections 4.3.2, 4.3.3 and 4.3.4.

4.3.2 Semi-semantic POS (SSP) Tagset Evaluation

The detail and definition of the semi-semantic POS (SSP) tagset was already described in Section 3.3.2.1. The complete guidelines of the SSP tagging were given to students for annotation of sentences according to a procedure described in Section 4.3.1. The tagged sentences by the annotators were recorded in MS Excel in the form of a reliability data matrix. The SSP annotation evaluation has already been discussed in Section 4.2.2.3. The reliability data matrix for a sample of sentence is depicted in Table 4.6 and discussed there. The coming Sections 4.3.3 and 4.3.4 for the SSS and F annotation evaluation will contain further relevant annotation examples. From the reliability data matrix, values by tokens matrix is obtained as given in Table 4.7. The α coefficient is computed according to the formula given in Equation 4.14. Different variables of numerator and denominator of equation are computed and displayed in the lower part of Table 4.7. The value of the α coefficient obtained is 0.964 for SSP tagging of anno-

Table 4.8: Annotators SSP tags distribution and confusion

POS Tags (Initial version)	Frequency	Identical	Different	Different/Confused/Disagreed Tags	%Accuracy
ADJ	270	265	5	BLANK	98.15
ADV	30	30	0		100.00
ADV.DEG	10	10	0		100.00
ADV.MNR	20	20	0		100.00
ADV.NEG	10	10	0		100.00
ADV.SPT	20	20	0		100.00
ADV.TMP	70	70	0		100.00
C.CORD	30	30	0		100.00
CM	630	630	0		100.00
DATE.Y.CAL	20	8	12	DATE.Y	40.00
DIA.IZF	90	39	51	DIA	43.33
DIA.PESH	10	3	7	DIA	30.00
KER	150	71	79	CM	47.33
N	1010	1005	5	BLANK, N.SPT	99.50
N.PROP	80	80	0		100.00
N.PROP.SPT	10	10	0		100.00
N.SPT	60	56	4	N	93.33
N.TMP	60	60	0		100.00
P.DEM	50	50	0		100.00
P.INDF	10	10	0		100.00
P.PERS	200	197	3	P.PRO	98.50
P.POSS	10	8	2	P.PERS	80.00
P.POSS.REF	10	10	0		100.00
P.REL	10	10	0		100.00
POSTP	60	60	0		100.00
POSTP.CMP	30	25	5	POSTP	83.33
POSTP.SPT	20	20	0		100.00
POSTP.TMP	20	20	0		100.00
PREP	20	20	0		100.00
PT	20	9	11	PT.INTF	45.00
PT.INTF	30	13	17	PT	43.33
Q	30	30	0		100.00
Q.CARD	210	210	0		100.00
Q.FRAC	20	20	0		100.00
Q.ORD	40	36	4	Q.CORD, Q.CARD	90.00
QW	50	50	0		100.00
U	30	30	0		100.00
V.COP.IMPERF	20	20	0		100.00
V.COP.PERF	10	10	0		100.00
V.COP.ROOT	20	20	0		100.00
V.COP.TENS.PRES	110	57	53	V.COP.PRES, V.LIGHT.TENS.PRES, VAUX.TENS.PRES	51.82
V.INF	10	10	0		100.00
V.LIGHT.IMPERF	40	37	3	V.LIGHTV.IMPERF	92.50
V.LIGHT.KER	50	21	29	V.LIGHT.ROOT	42.00
V.LIGHT.PERF	40	36	4	V.LIGHTV.PERF, V.PERF	90.00
V.LIGHT.ROOT	30	30	0		100.00
V.LIGHT.TB.ROOT	30	13	17	V.LIGHT.ROOT	43.33
V.LIGHTV.PERF	60	54	6	V.LIGHT.PERF	90.00
V.LIGHTV.PROG.IMPERF	10	5	5	V.LIGHTV.IMPERF	50.00

V.PERF	200	189	11	VAUX.PASS.PERF	94.50
V.ROOT	20	20	0		100.00
V.TENS.PRES	50	21	29	V.PRES	42.00
VAUX.IMPERF	20	17	3	V.PASS.IMPERF	85.00
VAUX.LIGHTV.TENS.PRES	10	4	6	VAUX.PRES	40.00
VAUX.MOD.PERF	30	30	0		100.00
VAUX.PASS.PERF	40	40	0		100.00
VAUX.PROG.PERF	20	9	11	VAUX.PROG, V.COP.PERF	45.00
VAUX.TENS.FUTR	20	10	10	VAUX.FUTR	50.00
VAUX.TENS.PAST	10	3	7	VAUX.PAST	30.00
VAUX.TENS.PRES	220	101	119	VAUX.PRES	45.91

tators. The value of α obtained lies in the category of perfect agreement according to Krippendorff. As a perfect agreement of 0.964 has been found in case of SSP annotation, which means that the SSP annotation guidelines are reliable. The error analysis and discussion of issues related to SSP tag set evaluation is given in Section 4.4.1.

The format of data evaluation in the Krippendorff's α is different from the data displayed in Table 4.8, however, a sample of annotators SSP tags distribution and confusion is displayed in Table 4.8. The annotated data of 100 sentences that were given to the annotators contained 1281 tokens, from which the data of 904 tokens is presented. The rest of the tokens have accuracy almost more than 90% due to which they are not depicted. It is attempted almost to show the tags of those tokens on which the annotators were remained confused or disagreed. The tags used in the initial version of the URDU.KON-TB treebank are displayed in the first column of the table. Adjective (ADJ) appeared 54 times in the sentences, which were then annotated by 5 annotators. The frequency of adjective annotation is depicted in the second column of the table after multiplying 54 with 5 number of annotators. It concludes 270 times of annotation for adjective. Among the 270 annotations of ADJ, annotators were remained 265 times in agreement or the annotators assigned 265 times the same/identical tag ADJ. The number of times the annotators remained disagreed or confused is mentioned in the different column. Similarly, the different or confused or the disagreed tags used by the annotators are depicted in the next column. Finally, the percentage accuracy of each tag in the first column of the table is calculated by dividing the values in the identical and the frequency columns.

The SSP tags in the initial version of the URDU.KON-TB treebank, which are correctly annotated and have 100% accuracy include ADV and ADV with its semantic labels for adverbs, coordination conjunctions with C.CORD, case markers with CM, N.PROP, N.PROP.SPT and N.TMP for proper nouns, spatial proper nouns and

temporal nouns, P.DEM and P.INDF for demonstrative and indefinite pronouns, etc. The tags contained less than or equal to 50% accuracy include tense auxiliaries e.g. VAUX.TENS.PRES mostly annotated differently with VAUX.PRES, progressive auxiliary e.g. VAUX.PROG.PERF annotated differently with VAUX.PROG and its copular behavior with V.COP.PERF, KER as a light verb e.g. V.LIGHT.KER annotated with V.LIGHT.ROOT, diacritics e.g. DIA.IZF with DIA only, etc. Annotation of some tokens with tags was left by the annotators represented with BLANK in the column 'different/confused/disagreed tags' for each tag in the first column of the table.

The error analysis and evaluation of tags was performed on the basis of this data depicted in Table 4.8. First, the tags with less or equal to 50% of accuracy are revised with annotators decisions e.g. DATE.Y.CAL, PT.INTF, V.LIGHT.TB.ROOT, etc., and second are the tags with accuracy a little more than 50% but they have common confused pairs like V.COP.TENS.PRES and VAUX.TENS.PRES modified to V.COP.PRES and VAUX.PRES, respectively as can be seen in the table. The detailed discussion on error analysis and evaluation of the SSP annotation is presented in Section 4.4.1.

4.3.3 Semi-semantic Syntactic (SSS) Tagset Evaluation

The details of semi-semantic syntactic (SSS) tag set were described in Section 3.3.2.2. The tag set along with the SSS annotation guidelines were given to the annotators for annotation. Their annotated data was recorded and evaluated as discussed in Section 4.3.2. The difference between the SSP annotation and the SSS annotation is the unit values (phrases) and cell values (phrase level tags) in the reliability data matrix as can be seen in Table 4.9 for the example sentence 4.15. The syntactic tagging of *us nE* is KP.ERG (ergative case phrase) due to *nE* ergative case marker, *bacE kO* is KP.ACC due to an accusative case marker *kO* and the last perfective verb *uTHAyA* lies in VCMAIN phrase. The annotators tagged the phrases of this example sentence according to their own understanding, which are given in the following SSS reliability data matrix.

(4.15) *us=nE* *bacE=kO* *uTHAyA*
KP.ERG KP.ACC VCMAIN
'He picked up the child'

As the units can be the combination of tokens due to phrases in the SSS reliability data matrix in contrast to the SSP reliability data matrix given in Table 4.6. So, the units can be represented by the gold-standard phrase/syntactic tags or with the combination of tokens in phrases as depicted in Table 4.9. The cell values in Table 4.9 are

Table 4.9: SSS Reliability Data Matrix

Annotators \ Syntactic Units	us nE	bacE kO	uTHAyA
$A1$	KP.ERG	KP	VCMAIN
$A2$	NP.ERG	KP.ACC	VCMAIN
$A3$	KP.ERG	KP.ACC	VCMAIN
$A4$	KP	KP	VCMAIN
$A5$	NP.ERG	KP.ACC	VCMAIN

the phrase/syntactic tags assigned to the respective units by the annotators. These phrase/syntactic tags are basically the representation of subtrees made by the annotators during their annotation. The root S level attachment of annotator's annotation is not recorded because it is true for all sentences. The unit by value matrix obtained through Table 4.9 is given in Table 4.10. The calculation in Table 4.10 is similar to

Table 4.10: A values by units (phrases) matrix

A-values \ Syntactic Units	us nE	bacE kO	uTHAyA	Sum	$n_{.1} * (n_{.2} : n_{.5})$	$n_{.2} * (n_{.3} : n_{.5})$	$n_{.3} * (n_{.4} : n_{.5})$	$n_{.4} * n_{.5}$
KP	1	2	0	3 ($n_{.1}$)				
KP.ACC	0	3	0	3 ($n_{.2}$)	9			
KP.ERG	2	0	0	2 ($n_{.3}$)	6	6		
NP.ERG	2	0	0	2 ($n_{.4}$)	6	6	4	
VCMAIN	0	0	5	5 ($n_{.5}$)	15	15	10	10
Sum	5($n_{1.}$)	5($n_{2.}$)	5($n_{3.}$)	**15**($n..$)	36	27	14	10
$n.. - 1$	14							
$n_{tp}n_{tq}$	8	6	0					
$n_{t.} - 1$	4	4	4					
$\frac{n_{tp}n_{tq}}{n_{t.}-1}$	2	1.5	0					
$\sum_t \frac{1}{n_{t.}-1} \sum_p \sum_{q>p} n_{tp}n_{tq}\ {}_{metric}\delta^2_{pq}$	3.5							
$\sum_p \sum_{q>p} n_{.p}n_{.q}\ {}_{metric}\delta^2_{pq}$	87							

Table 4.7 in Section 4.3.2. By putting the values calculated in Table 4.10 in Equation 4.14, we will obtain a value of α as 0.437 for the example sentence.

The method of calculation described is applied for evaluation of the SSS annotated data of annotators and the value of Krippendorff's α obtained in case of SSS annotation is 0.817, which lies in an acceptable category according to the Krippendorff discussed in

Table 4.11: Annotators SSS tags distribution and confusion

Syntactic Tags (Initial version)	Frequency	Identical	Different	Different/Confused/Disagreed Tags	%Accuracy
ADJP	70	70	0		100.00
ADVP	70	70	0		100.00
ADVP.MNR	20	11	9	ADVP	55.00
ADVP.TMP	40	27	13	ADVP	67.50
CL	45	23	22	VP, PP	51.11
CL.KER	150	71	79	KP, KP.POSS	47.33
CP	20	10	10	CORDP, BLANK	50.00
CP.NOM	10	5	5	CORDP.NOM, BLANK	50.00
KP	310	230	80	KP.POSS, KP.OBL, KP.MODF, KP.SPT, KP.TMP, KP.NOM	74.19
KP.ACC	40	34	6	KP	85.00
KP.DAT	10	10	0		100.00
KP.ERG	70	59	11	KP, NP.ERG	84.29
KP.INST	40	18	22	KP, KP.OBL, KP.POSS.INST, KP.POSS.INST.MODF, KP.INST.MODF	45.00
KP.POSS	130	101	29	KP	77.69
KP.SPT	10	6	4	KP, KP.POSS.SPT	60.00
KP.TMP	10	4	6	KP.POSS.TMP	40.00
KPQ.POSS	10	10	0		100.00
NP	360	343	17	N, NP.NOM, NP-TMP, BLANK,	95.28
NP.ACC	10	8	2	NP	80.00
NP.NOM	510	447	63	KP.SUB, NP, NP.NOM.SUB, NP.SUB, NP.NOM.PLINK	87.65
NP.REP.NOM	10	4	6	NP.NOM	40.00
NPQ	20	10	10	NP	50.00
NPQ.ACC	10	10	0		100.00
NPQ.NOM	20	10	10	NP.NOM	50.00
PP	120	113	7	QP, PP.SPT, PP.TMP, CL	94.17
PP.CMP	30	21	9	PP, PP-CMP	70.00
QP	60	60	0		100.00
UP	30	30	0		100.00
VCMAIN	530	530	0		100.00
VCP	190	139	51	BLANK	73.16
VIP	10	10	0		100.00
VP	80	65	15	CL.KER, CL	81.25

Section 4.2.2.3. However, it would be better if the value of α would become greater or equal to 0.900. It means that the reliability of the SSS annotation and the agreement among annotators is 0.817 and improvements can be made in this regard. The SSS tags were revised as per evaluation results and feedback obtained during this training course of annotation. The error analysis and discussion of issues related to SSS evaluation are presented in Section 4.4.2.

A sample of annotators SSS tags distribution and confusion is displayed in Table 4.11. The annotated data of 100 sentences that were given to the annotators contained 854 syntactic units, from which the data of 609 syntactic units is presented. The rest of the syntactic units have accuracy between 70% to 100% due to which they are not depicted. It is almost attempted to show the syntactic tags on which the annotators were remained confused or disagreed. The syntactic tags used in the initial version of the URDU.KON-TB treebank are displayed in the first column of the table.

Adverb phrase with manner concept (ADVP.MNR) appeared 4 times in the sentences, which were then annotated by 5 annotators. The frequency of ADVP.MNR annotation is depicted in the second column of the table after multiplying 4 with 5 number of annotators. It concludes 20 times of annotation for ADVP.MNR. Among the 20 annotations of ADVP.MNR, annotators were remained 11 times in agreement or the annotators assigned 11 times the same/identical tag ADVP.MNR. The number of times the annotators remained disagreed or confused is mentioned in the 'different' column as 9. Similarly, the different or confused or the disagreed tags used by the annotators are depicted in the next column as ADVP. Finally, the percentage accuracy of each tag in the first column of the table is calculated by dividing the values in the identical and the frequency columns.

The SSS tags in the initial version of the URDU.KON-TB treebank, which are correctly annotated and have 100% accuracy include ADJP for adjective phrase without its semantic labels, ADVP without its semantic labels for adverb phrase, KP.DAT for case phrase with dative case marker, KPQ.POSS for question case phrase with possessive case marker, NPQ.ACC for question noun phrase with accusative case marker, etc. The tags contained less than or equal to 50% accuracy include coordination phrase e.g. CP annotated differently with CORDP or left blank, coordination phrase with nominative case e.g. CP.NOM annotated differently with CORDP.NOM or left blank, instrumental case phrase e.g. KP.INST confused with KP, KP.OBL, KP.POSS.INST, KP.POSS.INST.MODF, KP.INST.MODF. Similarly, temporal case phrase KP.TMP annotated differently with KP.POSS.TMP, etc.

The error analysis and evaluation of tags was performed on the basis of this data depicted in Table 4.11. First, the tags with less than or equal to 50% of accuracy are revised e.g. KP.INST, KP.TMP, NP.REP.NOM, etc., and second are the tags with confusion in assigning of the semantic labels at syntactic level with '.' or '-' like ADVP.MNR and ADVP.TMP modified to ADVP-MNR and ADVP-TMP, respectively as can be seen in the table. Due to this modification, semantic labels at syntactic level will always be functional and annotated with '-' . Hence reduced the SSS tag set for the URDU.KON-TB treebank. Third is the case of those tags which have the low accuracy and a reason to modify, for example CP and CP.NOM are modified to CORDP and CORDP.NOM due to existing well known complementizer phrase CP and similarly, the CL.KER clause tag is reduced to CL due to removal of KER POS tag at the SSP annotation after revision. The detailed discussion on error analysis and evaluation of the SSS annotation is presented in Section 4.4.2.

4.3.4 Functional (F) Tagset Evaluation

This section deals with the evaluation process of the functional (F) annotation and its related issues. An updated version of the functional tag set developed is described in Section 3.3.2.3. The recording and evaluation process of functional annotated data is again the same as described in Sections 4.3.3 and 4.3.2. The functional tag set includes major grammatical relations as subject (SUB), object (OBJ), secondary object (OBJ2), oblique (OBL), predicate link (PLINK)[1] and modifier (MODF) along with other semantic, clause and miscellaneous labelings discussed in Section 3.3.2.3. The reliability data matrix for evaluation of the same example sentence 4.15 is thus constructed and displayed in Table 4.12. The difference between the reliability data matrix for SSS and

Table 4.12: Functional Reliability Data Matrix

Arguments / Annotators	us nE	bacE kO
*A*1	-SUB	-OBJ
*A*2	-ERG-SUB	-ACC-OBJ
*A*3	-SUB	-OBJ
*A*4	-SUB	-OBJ
*A*5	-ERG-SUB	-ACC-OBJ

F annotation is the unit columns (arguments of a sentence) and the cell values (tags for arguments) in table. The units can be labelled with gold standard functional tags e.g. SUB, OBJ, etc. or with the candidate tokens *us nE* 'he' for subject, *bacE kO* 'child' object, etc. as depicted in Table 4.12. In the example sentence, the predicate *uTHAyA* 'picked up' has the arguments subject (SUB) *us nE* and object (OBJ) *bacE kO* according to the gold standard. The dot '.' is used to represent a syntactical hierarchy of tags and '-' is used to represent the functional hierarchy of tags at syntactic level of annotation except the POS annotation, in which only '.' is used to represent all types of hierarchies. However, the annotators did not pay too much attention to this distinction of '.' and '-' and they mixed up this distinction during their annotation. Some of the annotators tagged KP.ERG-SUB as KP-ERG-SUB and similarly KP.ACC-OBJ as KP-ACC-OBJ. Due to this element, the data recorded in the reliability matrix in Table 4.12 contains -ERG-SUB and -ACC-OBJ in respective argument columns.

[1]The predicate and the predicate link are two different terms used in linguistics. The predicate is normally used for the main verb of the sentence and the predicate link is the argument linked with the subject of the sentence through the copula verb.

Table 4.13: A values by units (arguments) matrix

Arg-values \ Arguments	us nE	bacE kO	Sum	$n_{.1} * (n_{.2} : n_{.4})$	$n_{.2} * (n_{.3} : n_{.4})$	$n_{.3} * n_{.4}$
-ACC-OBJ	0	2	2 ($n_{.1}$)			
-ERG-SUB	2	0	2 ($n_{.2}$)	4		
-OBJ	0	3	3 ($n_{.3}$)	6	6	
-SUB	3	0	3 ($n_{.4}$)	6	6	9
Sum	$5(n_{1.})$	$5(n_{2.})$	**10**$(n..)$	16	12	9
$n.. - 1$	9					
$n_{tp}n_{tq}$	6	6				
$n_{t.} - 1$	4	4				
$\frac{n_{tp}n_{tq}}{n_{t.}-1}$	1.5	1.5				
$\sum_t \frac{1}{n_{t.}-1} \sum_p \sum_{q>p}$ $n_{tp}n_{tq}\ _{metric}\delta^2_{pq}$	3					
$\sum_p \sum_{q>p} n_{.p}n_{.q}$ $_{metric}\delta^2_{pq}$	37					

A values by units matrix obtained through the reliability data matrix is given in Table 4.13, which indicates that two of five annotators assigned the tag -ERG-SUB to candidate tokens *us nE* and three of five assigned the tag -SUB to the same candidate tokens. Similarly, for another combination of candidate tokens *bacE kO*, two annotators assigned -ACC-OBJ tag and three of them assigned an -OBJ tag. The remaining calculation in columns and rows of table is similar to Section 4.3.2 and 4.3.3. By putting the values of Table 4.13 in Equation 4.14, the α value obtained for this example sentence is 0.270, which shows a high disagreement among annotators.

The method described above was used to evaluate the functional annotated data of our annotators. The α value obtained through the evaluation of functional annotated data is 0.806, which lies in an acceptable level of agreement according to the Krippendorff (2004) recommendations. It can be greater than 0.900, which will be achieved surely in future, when the process will be repeated with amendments. The issues faced during and after this task of evaluation are amended and discussed in Section 4.4.3.

A sample of annotators F tags distribution and confusion is displayed in Table 4.14.

Table 4.14: Annotators F tags distribution and confusion

Functional Tags (Initial version)	Frequency	Identical	Different	Different/Confused/Disagreed Tags	%Accuracy
-CMP	20	13	7	BLANK	65.00
-COND	20	9	11	BLANK	45.00
-INST-MODF	10	4	6	-MODF, .INST-MODF	40.00
-INST-OBL	20	11	9	-OBL, BLANK, .INST-OBL	55.00
-INST-PLINK	10	4	6	-PLINK	40.00
-MNR-MODF	10	2	8	-MODF, BLANK, .MNR-MODF	20.00
-MODF	120	120	0		100.00
-OBJ	140	127	13	-ACC-OBJ, -NOM-OBJ, BLANK	90.71
-OBL	60	58	2	-POSS-OBL	96.67
-PLINK	130	125	5	-NOM-PLINK,	96.15
-SPT	10	7	3	BLANK	70.00
-SPT-MODF	40	35	5	-MODF, .SPT-MODF	87.50
-SPT-OBL	10	6	4	-OBL	60.00
-SUB	510	482	28	-ACC-SUB, -DAT-SUB, -ERG-SUB, -NOM, BLANK, -TMP-SUB	94.51
-TMP	20	17	3	BLANK, .TMP	85.00
-TMP-MODF	100	55	45	MODF, BLANK, .TMP.MODF	55.00
-TMP-SUB	10	10	0		100.00

The annotated data of 100 sentences that were given to the annotators contained 387 functional units, from which the data of 248 functional units is presented. The rest of the functional units have accuracy between 70% to 100% due to which they are not depicted. It is almost attempted to show the functional tags on which the annotators were remained confused or disagreed. The functional tags used in the initial version of the URDU.KON-TB treebank are displayed in the first column of the table. A label -CMP for comparative concept appeared 4 times in the sentences, which were then annotated by 5 annotators. The frequency of -CMP annotation is depicted in the second column of the table after multiplying 4 with 5 number of annotators. It concludes 20 times of annotation for -CMP. Among the 20 annotations of -CMP, annotators were remained 13 times in agreement or the annotators assigned 13 times the same/identical tag -CMP. The number of times the annotators remained disagreed or confused is mentioned in the 'different' column as 7. Similarly, the different or confused or the disagreed tags used by the annotators are depicted in the next column as BLANK, which means they didn't annotate this functional unit. Finally, the percentage accuracy of each tag in the first column of the table is calculated by dividing the values in the identical and the frequency columns.

The F tags in the initial version of the URDU.KON-TB treebank, which are correctly annotated and have 100% accuracy include -MODF for modifiers and -TMP-SUB for subjects having a temporal reading. The tags contained less than or equal to 50% accuracy include -COND for conditional clause, -INST-MODF for instrumental modi-

fier, -INST-PLINK for instrumental predicate link, etc.

The error analysis and evaluation of tags was performed on the basis of this data depicted in Table 4.14. First, the tags with less than or equal to 50% of accuracy are revised e.g. -COND is removed and second are the tags with confusion in adding to the syntactic level with '.' or '-' like the confusion existed in '.' usage in the annotation of -INST-MODF, -INST-PLINK, -MNR-MODF, etc, is now fixed by introducing '-' for all functional labels to add with syntactic level tags as can be seen in the respective columns of the table. Due to this modification, semantic labels at syntactic level will always be functional and added with '-' . Third is the case of tags which have the low accuracy and the reason to modify, for example predicate link PRD is modified to PLINK due to ambiguity in tag as PRD apparently denotes predicate but not predicate link. The detailed discussion on error analysis and evaluation of the SSS annotation is presented in Section 4.4.3.

4.4 Error Analysis and Discussion

4.4.1 SSP Analysis & Evaluation

The part of speech tags proposed initially (Abbas, 2012) needed to be revised after this exercise of annotation evaluation. It is difficult to discuss all the issues in detail and this section provides a concise summary.

1. There are two types of calendars in Urdu. One is the Isvi (AD) calendar and the other is the Hijri (AH) calendar. The symbols ء and ھ are used to denote the Isvi and Hijri calendar respectively. These symbols always reside along with the year number e.g. ۱۴۳۵ھ '1435 AH' and ۲۰۱۳ ء '2013 AD'. In a modern Urdu text, the well known numerals called Hindu-Arabic or Arabic-Hindu numerals (Kunitzsch, 2003) are commonly in use like ء 2013 or ھ1435. These symbols are some times explicitly mentioned in the complete word form e.g. عیسوی 'Isvi' or ھجری 'Hijri'. Another calendar called Solar Hijri (SH) calendar has been wiped out completely since decades and even the Hijri (AH) calendar only exists in religious literature. Moreover, in modern Urdu texts, only the year number e.g., 2013 according to AD is written in the text without any Isvi or Hijri symbol.

   In the initial version of the URDU.KON-TB treebank, a DATE tag represented dates was divided into the DATE.M, DATE.D, DATE.Y and the DATE.Y.CAL to represent months, days, years and a year with a calendar symbol, respectively.

During evaluation, it became clear that the annotators did not differentiate between DATE.Y and DATE.Y.CAL tags. Most likely, they used DATE.Y instead of DATE.Y.CAL tag for the word having a calendar symbol at the end. Due to this behavior of annotators, the hierarchy of the tag DATE.Y.CAL was limited to DATE.Y. The .CAL for calendar representation has been removed completely in the updated tag set shown in Table 3.3.

2. The different forms of the verb *hO* 'be/become' had a tag TB for 'to be' in the initial version. The verb *ho* in Figure 4.2(a, b) would had been tagged as V.LIGHT.TB.ROOT and V.TB.IMPERF respectively. The other possibilities of this verb include V.TB.PERF, V.COP.TB.ROOT, V.TB.ROOT, VAUX.TB.PERF, etc., due to the hierarchical design of tagging and morphology. During the annotation evaluation course, it was found that the tag TB assigned to verb *hO* was confusing for the annotators. All of its morphological forms were already covered in ROOT (root form), INF (infinitive form), PERF (perfective form), IMPERF (imperfective form) and SUBTV (subjunctive form) tags, so it was decided to remove this distinction of the TB tag and then treated the different forms of the verb *hO* like other verbs presented in Figure 4.2(a, b).

3. The different forms of present and the past tense-auxiliaries *hE, tHA* 'is, was' were also the part of the verb-verb complex predicate (LIGHTV) formation in the initial version of the URDU.KON-TB treebank. Example 4.16 is given according to the initial version of the POS annotation, in which a present tense auxiliary temporally situates the complex predicate formation.

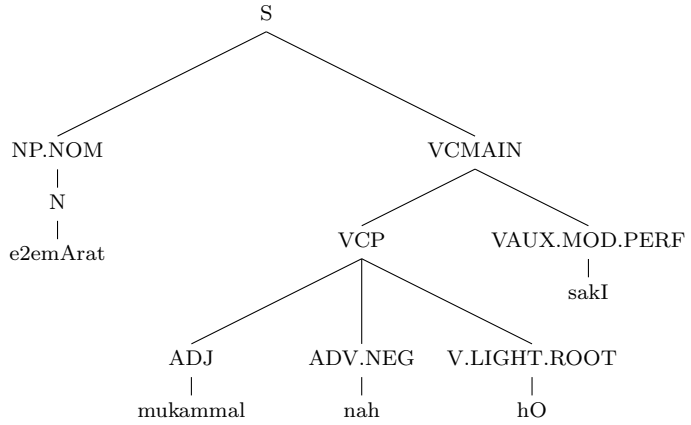
(4.16)

Us	*nE*	*kitAb*	*hAs3il*	*kI*	*hE*
P.PERS	CM	N	N	V.LIGHT.PERF	VAUX.LIGHTV.PRES

'He has got a book'

Here, *hAs3il kI* 'got' is a noun-verb complex predicate with *hAs3il* 'receiving' as a noun and *kI* 'did' as a light verb V.LIGHT.PERF with perfective morphology. The present tense auxiliary *hE* 'is' at the end of the sentence is considered to be the part of the complex predicate as VAUX.LIGHTV.PRES with the preceding V.LIGHT.PERF verb *kI*. The annotators response on these types of sentences has been found to be misleading. Most of the time, annotators ignored the annotation of LIGHTV and labelled the present tense auxiliary as VAUX.PRES like normal tense auxiliaries. In this situation, the rules for identifying the complex predicates (Butt, 2010) were revisited. After finding no sound proof of tense

(a). *e2emArat mukammal nah hO saki* 'The building could not be completed'



(b). *jab yahAN gHar hotE tHE* 'When, here would have been houses'

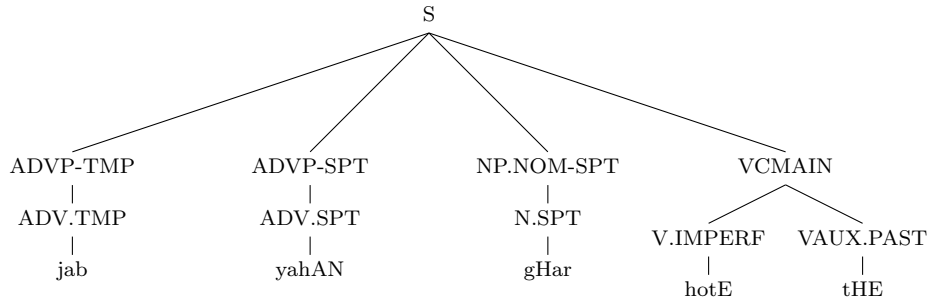


Figure 4.2: Verb hO 'be/become' after removal of the TB tag

auxiliaries as a light verb, the marking of tense auxiliaries as the light verbs was abandoned and updated.

4. A word *rahA/rahI* can be treated as a verb with a perfective morphology like in 4.17 and can also be treated as a continuous/progressive auxiliary like in 4.18. In the initial version, all the occurrences of this particular word *rahA/rahI* were annotated accumulatively as VAUX.PROG.PERF (an auxiliary with perfective and progressive forms at the same time). This annotated tag was found to be false because the word *rahA/rahI* can behave as a copula verb (Raza, 2011). Its identification was considered to be a complex task for annotators. However, the annotators performance is admirable in this concern. It is seen in the annotated sentences that 3 out 5 annotators performed very well with the type of sentences in 4.17. The response of annotators on these type of words is V.COP.PERF, which

means that the verb is acting as a copula verb with perfective morphology. It is a good distinction and can be verified in the literature. These reasons inspired the author to remove the accumulative tag VAUX.PROG.PERF and introduce a new tag for copula verbs as can be seen in 4.17.

(4.17) *h2AzrI* *kam* *rahI*
N ADJ V.COP.PERF
'The attendance remained less'

(4.18) *voh* *garI* *calA* *rahA* *hE*
P.PERS N V.PERF VAUX.PROG VAUX.PRES
'He is driving the vehicle'

5. Among the annotation of particles, two tokens remained controversial. These tokens were *bHI* 'too/also' and *hI* 'only'. The respective tags used for two tokens were PT.INTF (particle-intensifier) and PT (particle). The annotators failed to tag these tokens as intended. The evidence shows that the annotators interchanged the tags for the two tokens often. Hence, only PT.INTF is kept for both of the tokens *bHI* (بھی) and *hI* (ہی) in the updated tag set. The main POS tag PT has further subcategories as adjectival particle PT.ADJ, emphatic particle PT.EMP, intensifying particle PT.INTF and resultant particle PT.RESULT in the SSP tag set finally.

6. Currency names were not frequently annotated with the proposed tag N.CURR by the annotators. The annotators preferred the tag N for noun or N.PROP for proper noun. So, the currency tag was removed from the SSP tag set completely.

7. The two words *kar* 'do' and *kE* 'of' can come together and separately in a sentence. Their distant occurrence has a different syntactic and semantic definition as compared to close occurrence. Individually, *kar* is a verb with root form and *kE* is a possessive case marker. When they come together as *kar kE* 'after doing', then they provide a same reading of its another version like the postposition phrase *karnE kE ba2d* 'after doing'. This version may be the detailed version of the first one. Collectively, the *kar kE* identifies a clause boundary and a KER tag was assigned to *kE* in the initial version of the SSP tag set. There is no issue in the distant occurrence because the *kar* was tagged as a verb with root morphology V.ROOT and the *kE* was tagged as a possessive case marker CM. In close combination of *kar kE*, the *kar* had the tag V.ROOT, while the *kE* had the tag as KER in the initial version. The *kE* with KER tag then became the head word

and made a CL.KER clause at the syntactic level of annotation. Unfortunately, the annotators did not like this distinction too much and they treated the distant and the close cases of *kar* and *kE* with the same stick of tags as V.ROOT and CM, respectively. Due to this reason, the KER tag in the close combination is removed. Now the clause CL is readable only through the presence of tokens for the close combination.

8. It was seen in the annotated data that when tense auxiliaries e.g. VAUX.PRES come alone in the VCMAIN (main verb phrase) which is the case of tense auxiliaries behaving as a main verb in the sentence, specially in the case of copular construction, they assign V.PRES or V.COP.PRES tags in non-copular and copular constructions respectively. Similarly, the annotators did not show too much interest in the labeling of TENS (tense) tag before tenses according to the initial version. For example, the initial version VAUX.TENS.PRES was annotated as VAUX.PRES only. So, the observed trend of annotators on the tense auxiliaries has been applied in the annotation of the URDU.KON-TB treebank to make it more reliable.

9. Diacritics do not have the status of letters and are mostly placed above or below the letter. Like Arabic, the use of diacritics in Urdu script is not a routine, but some diacritics are still used in Urdu script writing (Haq, 1914; Siddiqui, 1977) which includes Zabar, Kasra, Pesh, Fatah, Tashdeed, Tanveen, etc. For example, the Zer-e-Izafat is very common phenomenon in Urdu script, which is an enclitic short vowel 'e' used to join two nouns or a noun and an adjective in a compound word (Bögel and Butt, 2013; Schmidt, 2013) as follows.

(4.19) آبِ حیَات

Ab=e *h2ayat*
water=IZF life
'Water of life'

(4.20) مغلِ اَعظم

mUGl=e *az4am*
mughal=IZF great
'Great Mughal'

Diacritics under the last letters بِ and لِ of the first words are the examples of Zer-e-Izafat. In the initial version of the URDU.KON-TB treebank, the only diacritic tags used were DIA.IZF and DIA.PESH for representing the Zer-e-Izafat and Pesh at the last letter of the first word in a compound word. However, some annotators did not follow the guidelines to tag these diacritics, even though the respective diacritics were present in the raw text provided. In this situation, the further division of DIA tag as IZF and PESH was removed, leaving the DIA tag only to represent all types of diacritics at the last letter of the first word in a compound word. The other diacritics in a word are considered as a normal text. The last letter diacritic is needed because it behaves as a joiner with the next word, which is useful to identify the sub-phrase or phrase boundary and thus helpful in automatic speech processing.

10. From the initial version of the URDU.KON-TB treebank, a tag V.HABT.IMPERF for verb and a tag VAUX.HABT.IMPERF for verb auxiliary were found to be redundant during the course of annotation evaluation. The reason of their redundancy was the HABT and the IMPERF tags existing for the same purpose. In Urdu imperfective morphology can be identified by *tA/tI/tEN/tE* at the end of a verb or auxiliary. Similarly, the reading of an habitual activity in Urdu can be identified through a verb auxiliary *rEhTA/rEhtI/rEhtEN/rEhtE* 'use to' as can be seen in the example (4.21). In the initial work, both the HABT and the IMPERF morphological feature for an auxiliary *rEhtE* was tagged as the VAUX.HABT.IMPERF in the single tag. This redundancy is removed by deleting the HABT tag completely from the SSP tag set as presented in 4.21.

(4.21)

voh	*Ek*	*dUsrE*	*sE*	*laRtE*	*rEhtE*
P.PERS	Q.CARD	Q.ORD	CM	V.IMPERF	VAUX.IMPERF

tHE
VAUX.PAST

'They were used to fight with each other.'

11. In the initial version of the POS annotation, some verb-verb compound complex predicates (Raza, 2011) were annotated as given in example 4.22 as follows.

(4.22)

gARIAN	*astESan*	*par*	*AtI*	*jAtI*	*hEN*
N	N.SPT	CM	V.IMPERF	V.LIGHTV.IMPERF	VAUX.PRES

'The vehicles come and go at the station'

The verb-verb compound complex predicate *AtI jAtI* 'come and go' is annotated as *AtI*/V.IMPERF *jAtI*/V.LIGHTV.IMPERF in the example 4.22. Some of the annotators argued about the presence of the light verb here in this construction after the submission of their annotated data. They commented that this sentence is the short form of two coordinated sentences presented in example (4.23), which can be further shortened as *gARIAN astESan par AtI aOr jAtI hEN* 'The vehicles come and go at the station'. This shortened form *AtI aOr jAtI* 'come and go' is not the case of compound complex predicates (Raza, 2011), which we adopted. Being in this situation, the tags are revised for this kind of formation and annotated as *AtI*/V.IMPERF *jAtI*/V.IMPERF.

(4.23) *gARIAN* *astESan* *par* *AtI* *hEN* *aOr* *gARIAN*
N N.SPT CM V.IMPERF VAUX.PRES C.CORD N
astESan *sE* *jAtI* *hEN*
N.SPT CM V.IMPERF VAUX.PRES
'The vehicles come at the station and the vehicles go from the station'

The discussion of issues related to the SSP tag set presented above provides a flavor of the problems faced during and after the annotation evaluation. Three out of the eleven issues numbered as 2,3 and 11 are related to complex predicates. To resolve and identify these complex predicates, a theory of complex predicates given by Butt (2010) has been adopted fully and outlined in clear steps in the annotation guidelines of the URDU.KON-TB treebank presented in Chapter 2. According to this, different combinations of light verb are possible. (i) A verb complex predicate is possible in a combination of a noun and a light verb e.g. *h2Asil*/N *kI*/V.LIGHT.PERF 'obtained'. (ii) A VCP can contain an adjective and a light verb e.g. *pAk*/ADJ *karna*/V.LIGHT.PERF 'to clean'. (iii) A VCP can be the combination of a root form of verb and a light verb e.g. *gir*/V.ROOT *paRA*/V.LIGHTV.PERF 'fell'. (iv) Causative possibility is the combination of a causative verb and a aspectual light verb e.g. *rulA*/V.PERF *dIyA*/V.LIGHTV.PERF (made someone cry). (v) A combination of a verb and a permissive verb e.g. *dEkHnE*/V.INF *dIyA*/V.LIGHTV.PERF (let someone to see) and others. In the initial version, the verb complex predicates theory given by Butt (2010) was expanded with some additional types of complex predicates given in Raza (2011) to provide rich information regarding complex predicates in one unit. The annotators did not show a tendency to annotate a variety of complex predicates in their annotation, due to which the guidelines given in Chapter 2 are revised and limited to the theory of complex predicate given by Butt (2010).

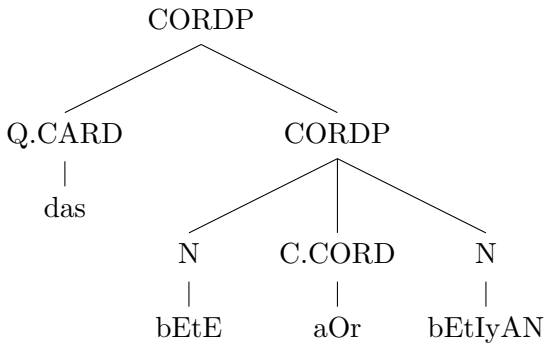


Figure 4.3: SSS annotation: a CORDP example

4.4.2 SSS Analysis & Evaluation

The initial version of the syntactic tag set proposed by Abbas (2012) was revised after the evaluation task of annotation. The revision based on the annotated data and feedback of the annotators consists of renewal or removal of tags from the initial version. A brief discussion of the issues is given as below.

1. The coordinating conjunction C.CORD caused the phrase to be tagged as CP (conjunction phrase) in the initial version. For linguists, CP is commonly used for a complementizer phrase, so ambiguity arose in the annotation of annotators. It has been found that some annotators did not consider the CP tag. They either avoided it or tagged it with other head words except coordinating conjunction in a phrase. To improve the agreement among annotators, a rule was introduced in the presence of C.CORD. This rule narrates that making of CORDP (coordinating phrase) is essential in the presence of C.CORD instead of CP. For example, a CORDP would be annotated as in Figure 4.3 for a phrase in 4.24.

   (4.24) *das* *bEtE* *aOr* *bEtIyAN*
   ten sons.Masc.Pl and daughters.Fem.Pl
   'Ten sons and daughters'

   Due to the existence of C.CORD *aOr*, the head phrase become CORDP in a subtree, which is acting again as a head phrase at the root node in the presence of Q.CARD *das*.

2. In the initial version of the annotation, the phrase could have the same nested phrase again with different tokens. This phenomenon was represented by a .REP

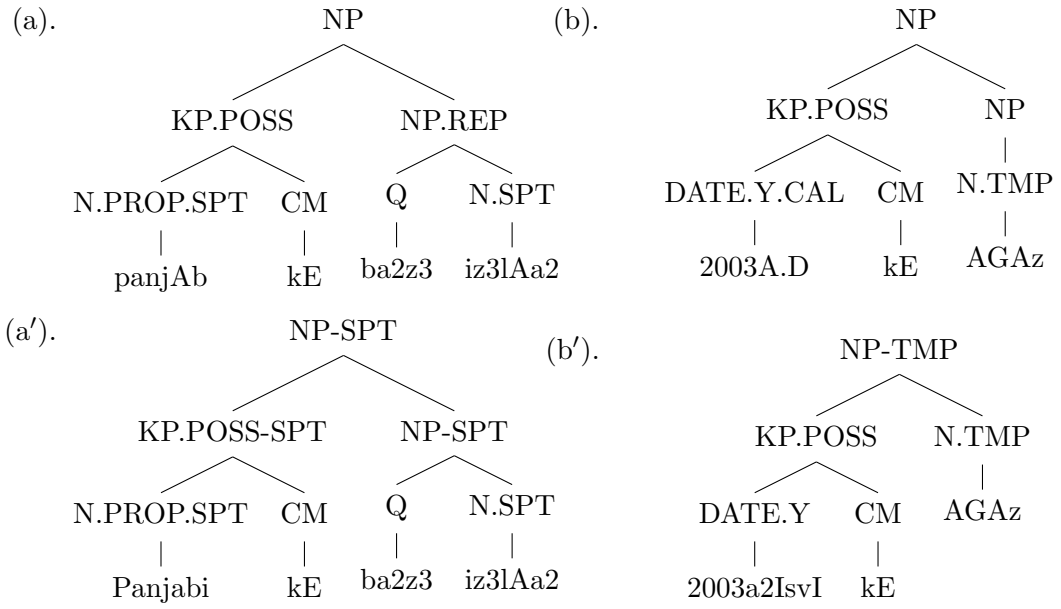


Figure 4.4: SSS annotation: with or without nested NP

tag to denote nested repeated phrase e.g. NP (noun phrase) can have other NP in it, which is then represented by NP.REP (repeated noun phrase) according to the initial version of the annotation. The annotators did not show the interest to annotate the same repeated nested phrases with .REP. To deal with this situation, the .REP rule was removed at the syntactic level of annotation. Now, if an NP contains another NP then it is tagged with a simple NP again and so on. Moreover, a nested NP is made only when a head noun exists with some other non-head token/tokens having a different POS as can be seen in Figure 4.4(a′,b′) except the particles PT. The particles are always attached flat in the subtree of a phrase, where they come and hence they have no effect on the syntactic structure. The difference of the initial and the revised annotation style is depicted in Figures 4.4(a-a′, b-b′) for the phrases given in 4.25 and 4.26.

(4.25) *panjAb=kE* *ba2z3* *iz3lAa2*
Punjab.Masc.Sg=Poss few districts.Masc.Pl
'Some districts of Punjab'

(4.26) *2003-a2IsvI=kE* *AGAz*
2003-A.D=Poss beginning
'Beginning of 2003 A.D'

Figure 4.4(a) is an example of the initial annotation style for 4.25. A repeated noun phrase NP.REP in it is updated in the new annotation style depicted in Figure 4.4(a′). Spatial information can traverse up at syntactic level along with the head word which is discussed in detail in Section 4.4.3. The Figure 4.4(b) has the old annotation style of nested NP. It contains a single head noun and the revised annotation style is given in Figure 4.4(b′).

3. The different readings of a sentence can cause the false identification of the syntactic tag. An example of a rhetoric question is given in 4.27.

(4.27) *jis=kE* *bacE* *ta2lim=sE* *meh2rUm* *hUN,*
whose=Poss children.Pl=Nom education=Inst deprive be.Pres
usE *bacON=kI* *nokrION=sE* *kiyA* *kAm* *hE*
he.Sg=Dat children.Pl=Poss jobs.Fem.Pl=Inst what work be.Pres.Sg
'Those whose children are deprived of education, what use do they have of their children jobs'

This sentence contains a rhetoric question 'What use do they have of their children jobs', which is not basically the actual asking. It is a conclusive statement with actual reading as 'they have no concern with their children's jobs' after the first part of the sentence 'Those whose children are deprived of education'. The sentence contains a part *kiyA kAm* 'What type of work', which was tagged with a question word and a noun as *kiyA*/QW and *kAm*/N respectively. The syntactic tag is annotated as a noun phrase having a question word (NPQ (QW kiyA) (N kAm)). In the initial version, only the QW was existed at the POS level for all the types of question words. The syntactic tag set had the SQ and SBARQ for a yes/no question sentence and a question subordinate clause, respectively. If a subordinating clause SBAR contained a question word then it was tagged with SBARQ. Similarly, if a sentence had a question with a yes/no concept, then the root S for the sentence contained another subroot as the SQ to differentiate the yes/no question sentences. These rules forced the subordinating clause SBAR of example 4.27 to be tagged as the SBARQ, which was a wrong solution because the subordinating clause has only a rhetorical question and it does not contain the actual asking. Moreover, the annotation of this kind of question phrase types e.g. (NPQ (QW kiyA) (N kAm)) was found to be annotated as simple noun

phrases with (NP (QW kiyA) (N kAm)) in the annotated data of annotators, which is due to different readings taken by the annotators.

To provide a clear distinction for making a question phrase, some rules were devised after considering the context before or after the question word in a sentence, which are as follows.

(a) If the question word is not alone then make the question phrase according to the last word in that phrase. If the last word is the case marker CM then the phrase would be the case phrase having a question as KPQ e.g. *kis/QW kE/CM* 'to whom/who', if the last word is a noun then the phrase would be an NP having a question as NPQ e.g. *kis/QW had/N* 'what limit', etc. The question phrase tags can be seen in the SSS tag set depicted in Table 3.5.

(b) If the question word is alone, then make the phrase according to the nature of the question word e.g. AVPQ (adverb phrase having a question) for *kab*/ADV 'when'. There are some situations, where one can fail to identify the nature of the question word, then apply the answer replacement test to identify the nature e.g. *tumhArA nAm kiyA hE* 'what is your name?', if you replace a 'John' with *kiyA* in a question sentence then it would be the answer to your question. So, *kiyA*/QW can be tagged as NPQ at the phrase level. However, it is pertinent to note that the need to test the nature of question word is seldom required.

(c) If all the rules above have been failed and the nature or the type of the question word could not be found then QWP (question word phrase) can be used finally.

4. In the initial version of the annotation, a clause CL.KER was annotated when two words *kar* 'do' and *kE* 'of' came together as *kar kE* 'after doing'. The discussion of their POS tags as V.ROOT for *kar* and KER for *kE* is given in Issue 7 of Section 4.4.1. It was seen in the initial annotated data that the different syntactic structures existed for the formation of CL.KER clause as can be seen in Figures 4.5. This effect also appeared in the annotation of the annotators during the annotation evaluation. The subfigures (a), (b) and (c) in Figure 4.5 are the different annotations of a clause *545 sE baRH kar* 'after being more than 545' in the initial annotation. Similarly, the subfigures (d), (e) and (f) in Figure 4.5 are the various annotations of another clause *rAbtah kar kE* 'after doing communication' in the same version. These different syntactic structures of the CL.KER clause without any rule made the annotators confused and they

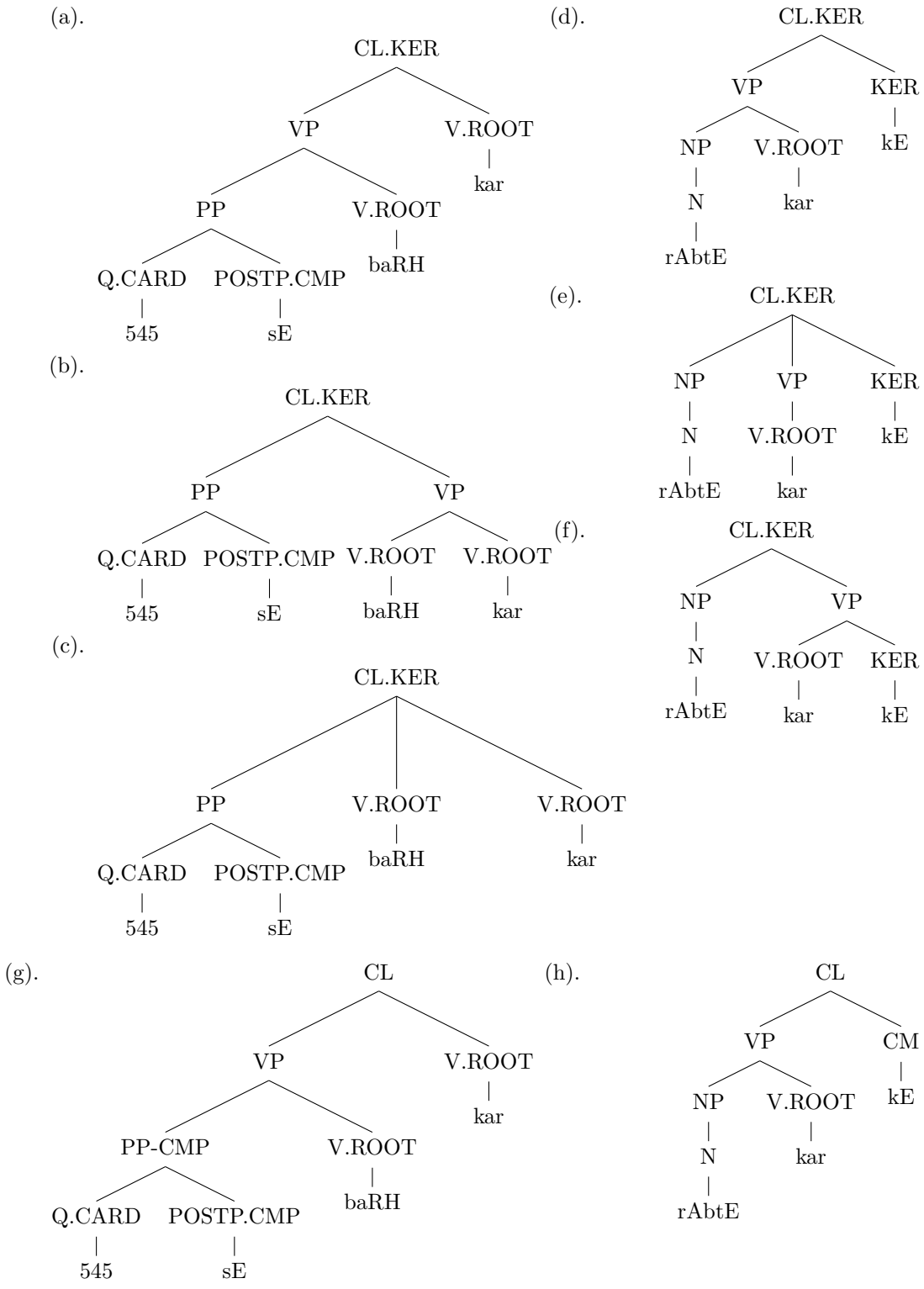


Figure 4.5: SSS annotation: a case of CL.KER clause

remained on building these confused formations continuously. To remove this structural ambiguity, (a) and (d) syntactic derivations in Figure 4.5 are finalized for head words *kar* and *kE* among the different available annotated options. The updated POS tag CM for the word *kE* is used, which was already discussed in Issue 7 of Section 4.4.1. The final syntactic structures for the formation of CL.KER clause as a clause CL is shown in (g) and (h) of Figure 4.5. According to which, the KER tag is totally removed at all positions, which was used in the initial annotation to distinguish a clause having *kar* or *kar kE* from other clauses. This distinction is removed and the clause with a special tag CL.KER is now treated like a simple clause with a tag CL (clause). Its syntactic formation is also fixed to make the verbal phrase first before the introduction of the CL as can be seen in Figures 4.5(g,h).

5. In the initial version, the verb phrase or the adposition phrase which behaved like a clause in the sentence were annotated as a clause CL as can be seen in 4.29 and 4.28, respectively. Their annotation of the verb phrase or the adposition phrase as CL (clause) finally was also confusing for the annotators. The annotators wanted to stop at the formation of the verb phrase or the adposition phrase and they did not want to annotate it further.

(4.28) *[bAp=kA sAyah sar=sE [UTH jAnE=kE*
[father.Masc.Sg=Poss shadow head=Inst [rise.Root go.Inf=Poss
ba2d]] vo bEcArI laRkIyAN har vaqt ANsU
after]] those poor.Fem girls.Fem.Pl each time tears
bahAtI rEhtI hEN
flow.Fem.Imperf remain.Fem.Imperf be.Pres.Pl
'[[After rising up] the father shadow from the heads (his death)], those poor girls are remained on flowing tears each time'

It was decided that the annotation of the adposition phrase alone will not be derived to CL finally as can be seen in Figure 4.6(b′) for the bracketed clause part of the sentence partially in 4.28. A half of the clause part *bAp=kA sAyah sar=sE* 'the father shadow from the heads' is the part of the clause but not displayed to remain concise. Similarly, the verb phrase acting as a clause with perfective morphology of verb *hO* 'be/become' will be made VP inside a VP and will not be projected as a clause CL finally as can be seen in Figure 4.6(a′) for the bracketed part of the sentence in 4.29.

(4.29) *mErE Is GEr-matvaqEa2 savAl=par vo [kucH*
my this unexpected question=Loc he.Masc.Sg [some
sOctE hUE] bOlA
think.Imperf be.Perf] speak.Perf.Masc.Sg
'On my unexpected question, he spoke [while thinking something]'

6. During the initial annotation, a VCP (verb phrase having a complex predicate) was constructed independently when a noun N or an adjective ADJ followed by a light verb V.LIGHT with some morphological form and the successive occurrence of the second light verb V.LIGHTV with some morphological form was annotated out of VCP in the VCMAIN as can be seen in Figure 4.7(a). But the annotators introduced the VCP formation for both the categories of complex predicates as can be seen in Figure 4.7(b). This was found to be a good annotation, through which we can explore all the types of complex predicates under the umbrella of VCP.

7. Verb auxiliaries like modal, passive and progressive were sometime annotated as the 2nd category of complex predicates (V.LIGHTV as discussed in previous issue) in the initial annotation, if they appear and behave like verb-verb (V-V) complex predicate. For example, *nikAl/V.ROOT liyA/V.LIGHTV.PERF* 'took out' is a V-V complex predicate according to the theory of Butt (1995), but if

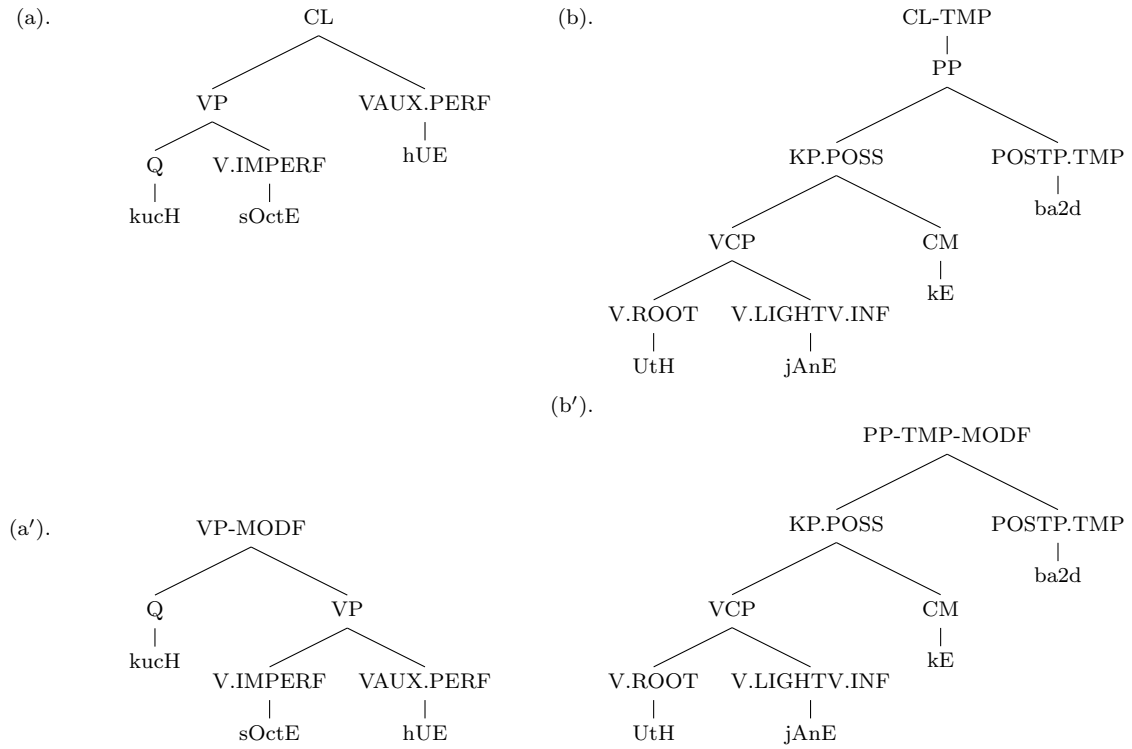
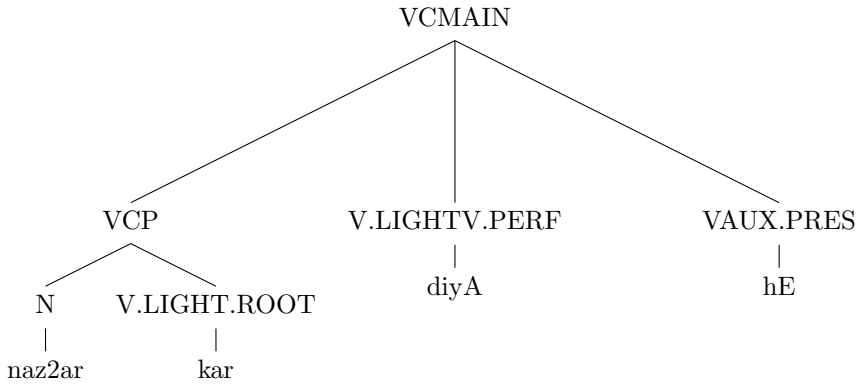


Figure 4.6: SSS annotation: a case of VP and PP clauses

(a).



(a′) .

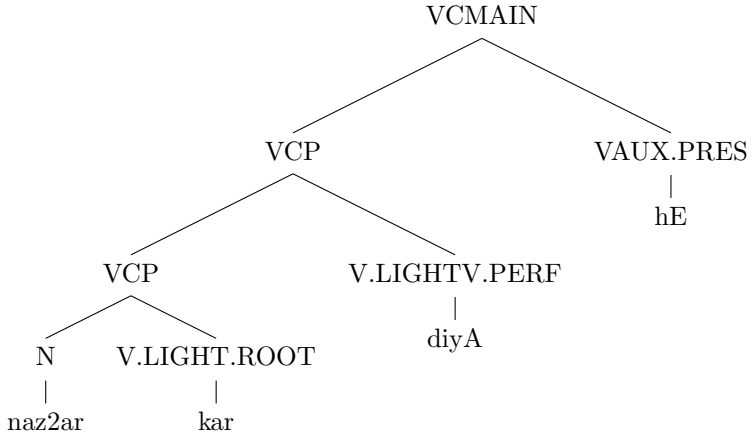


Figure 4.7: SSS annotation: a case of VCP

we replace the word *liyA* 'took' with a modal verb *sakA* 'could', then it would be *nikAl sakA* 'could take out' with same appearance as the complex predicate *nikAl liyA* but have a semantics of modality. Due to this reason, it was annotated as *nikAl/V.ROOT sakA/V.LIGHTV.MOD.PERF* in the initial version, which means a modal verb is acting as light verb with perfective morphology. This practice was also true for passive and progressive verb auxiliaries in the initial annotation. However, the annotators were unable to identify these type of complex predicates and they remained confused in their annotation. In this situation, we revised it according to theories designed for modal verbs by Abbas and Nabi Khan (2009); Bhatt et al. (2011). According to the revised rule, verb auxiliaries like modal (MOD), passive (PASS) and progressive (PROG) do not behave like the 2nd category of complex predicates and annotated as verb auxiliaries VAUX along with their function and morphology as can be seen in (a), (b) and (c) of Figure 4.8 for modal, progressive and passive auxiliaries in 4.30, 4.31 and 4.32, respectively.

(4.30) *tyAr* *kar* *saktA* *hE*
ready do.root can.imperf be.pres
'can prepare'

(4.31) *tyAr* *kar* *rahA* *hE*
ready do.root continue.imperf be.pres
'is preparing'

(4.32) *tyAr* *kiyA* *jatA* *hE*
ready did.perf go.imperf be.pres
'is prepared'

The issues presented is a compact version of syntactic problems. The syntactic structure helps in classifying the POS tags into syntactic units or phrases e.g., a noun can make a noun phrase, adjective can make an adjective phrase, etc. Similar to the POS information at the syntactic level, the semantic labeling lying at the SSP tag set can be projected at the syntactic level along with the functional information. This is discussed in the next section, which details the annotation evaluation of the functional tag set for the URDU.KON-TB treebank.

4.4.3 F Analysis & Evaluation

The issues faced during and after the training course of annotation evaluation were addressed and the initial version of the functional tag set in (Abbas, 2012) was revised

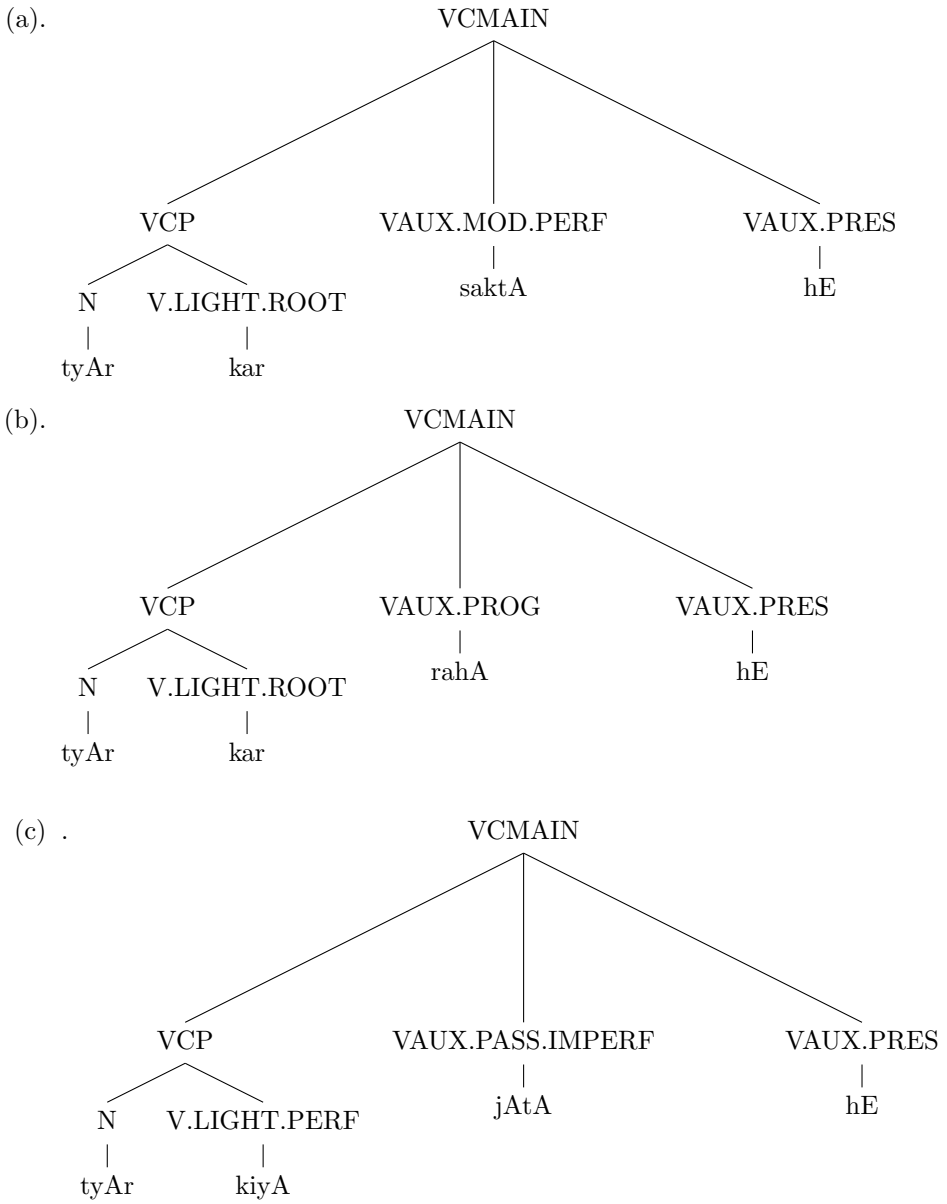


Figure 4.8: SSS annotation: a case of modal, progressive and passive verb auxiliaries

accordingly. In the revision, it was also attempted to remove ambiguities of those functional tags, which were causing a high degree of disagreement among annotators. The discussion of these issues is below:

1. According to the initial annotation, the semantic tags given in Table 3.6 like SPT (spatial), TMP (temporal), CMP (comparative), and INST (instrumental) could be tagged with '.' and '-' at the syntactic level of annotation. This was done by applying two methods. The first one was the design of the phrase, in which the word other than the case markers became the head word. If the head word contained a semantic label in its POS tag, then this semantic label was added to the syntactic tag of the phrase using the '-' symbol after it. Its annotation is presented in Figure 4.9(a′) for a sentence given in 4.33. The noun phrase contained a compound word *SAhrAhe rESam* 'Silk Road' acting as the head word. This compound word is a proper name of the trading route between Europe, Asia and the Middle East. It has a spatial SPT semantic label as a part of its POS tag N.SPT, which is then added/traversed to its syntactic tag NP using '-' according to its designed rule as NP-SPT. As this phrase is acting as the subject SUB of the sentence, which is then also added to the NP-SPT using the same symbol.

   (4.33) *SAhrAhe* *rESam* *lambI* *hE*
   road.Fem.Sg Silk.Masc=Nom long.Fem be.Pres.Sg
   'The Silk Road is long'

   The second method of design was that if the case marker became the head word of a phrase, then the semantic label lied at the POS tag of the dependent word or at the syntactic tag of the dependent phrase was added to the KP (case phrase) tag by using the '.' symbol after it. Its respective annotation is depicted in Figure 4.9(a) for a sentence given in 4.34. Due to locative/spatial case marker *par* 'on', the KP phrase is tagged as KP.SPT according to devised rule with additional grammatical tag PRD (predicate link) discussed in Section 3.3.2.3. This phrase KP.SPT-PRD is giving us information that the spatial case phrase is acting as a predicate link in the sentence.

   (4.34) *SAhrAhe* *rESam=par* *hajUm* *tHA*
   road.Fem.Sg Silk.Masc=Loc crowd.Masc=Nom be.Past.Masc.Sg
   'There was a crowd on the Silk Road'

   These distinct designs looked good but annotators failed to distinguish these and they mixed up the use of '.' and '-' in their annotation. In this situation, a simple

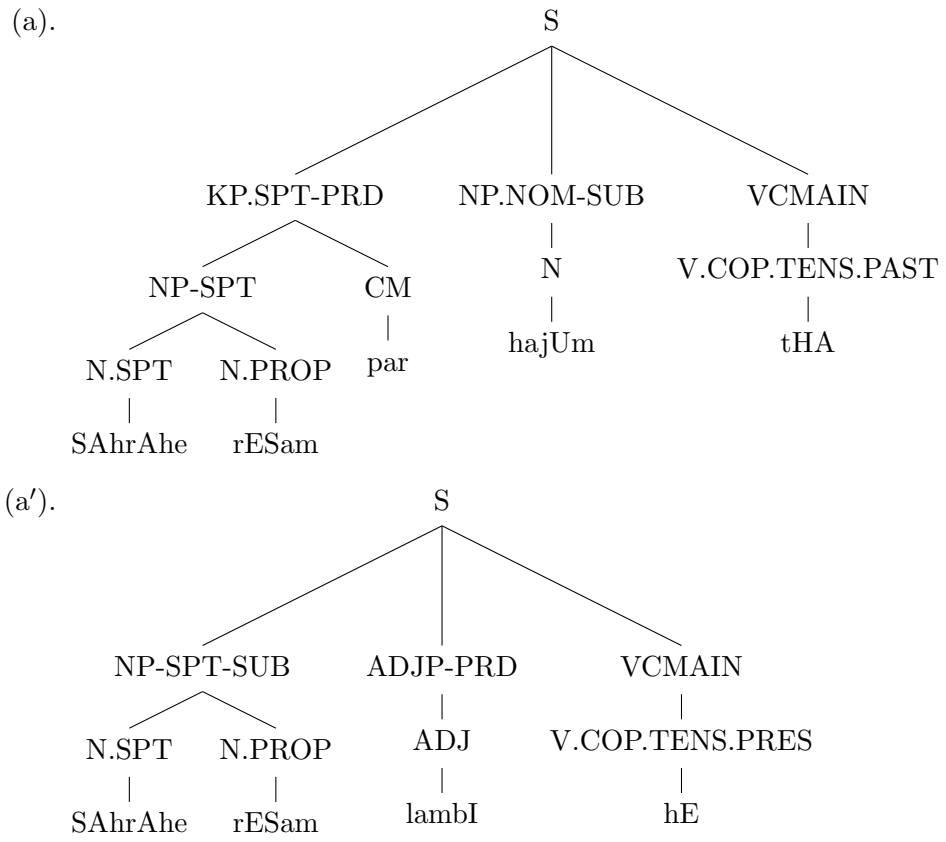


Figure 4.9: Initial F annotation: a case of SPT

revision was applied to hold the consistency. According to the new version of the annotation guidelines, the semantic labeling will always be annotated with '-' at syntactic level for both the cases discussed earlier. The updated version of Figure 4.9 is presented in Figure 4.10 according to new annotation guidelines.

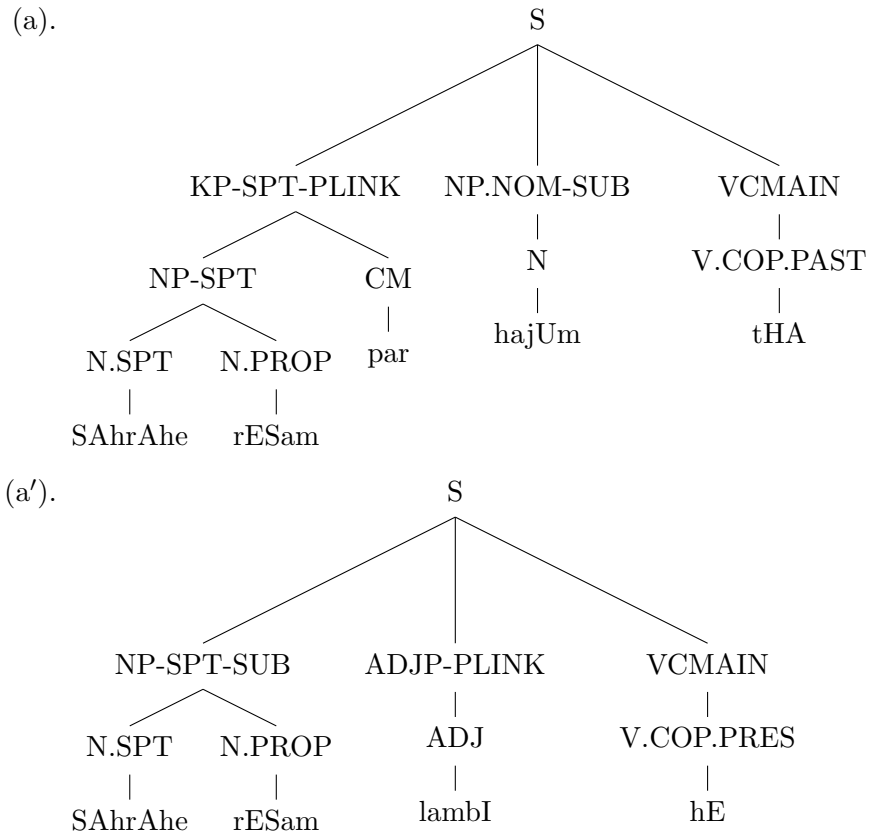


Figure 4.10: New F annotation: a case of SPT

2. During the training course of annotation, it was proposed to rename the tag PRD (predicate link) with a proper and a more common convention because the PRD abbreviates only predicate and not the predicate link. Both predicate and the predicate link are two different terms used in linguistics. The predicate is normally used for the main verb of the sentence and the predicate link is the argument linked with the subject of the sentence through the copula verb. It was decided to adopt the more conventional tag existing in Urdu LFG grammar of the ParGram[1] project to avoid the confusion. In that grammar, PREDLINK is

[1] `http://ling.uni-konstanz.de/pages/home/pargram_urdu/`

the convention used to represent predicate link of the sentence. We adopted the same convention with short version as PLINK depicted in Figure 4.10.

3. In the initial annotation, a functional tag COND was defined to represent conditional subordinate clause in a sentence as SBAR-COND. This was used if a conditional subordinate conjunction e.g., *agar* 'if' was in a subordinate clause. During the annotation evaluation, annotators did not use it. This trend prompted the author to remove the COND functional tag completely.

4. According to the initial version of the annotation, KP and PP phrases could be the object OBJ in a sentence rather than the oblique OBL argument, if they were behaving as an essential argument of the sentence predicate. The annotated trees of PP and KP are given in Figure 4.11 for the example sentences 4.35 and 4.36, respectively.

(4.35) *zindagI=kA* *safar* *mOt=kI* *t2araf* *muR*
life.Fem=Poss journey.Masc death.Fem=Poss towards turn.Root
jAtA *hE*
go.Imperf be.Pres
'The journey of the life turns towards the death'

(4.36) *vO* *Is* *makAn=mEN* *rEhtA* *hE*
He=Nom this house.Masc=Loc live.Imperf be.Pres
'He lives in this house'

In both of the sentences, the predicate is demanding further information, which is presented in PP and KP phrase respectively. The annotators were found to be confused as to mark in these phrases as oblique or object. According to Rizvi (2008), these phrases are the oblique arguments of sentences. The modified version of the PP and KP is depicted in Figure 4.12(a, a′).

5. The case marker *sE* has many functions (Butt and King, 2004; Khan, 2009; Mohanan, 1994) like instrumental, source expression, material, locative, temporal expression, etc. In the initial version, it was handled with the POS tag CM (case marker) and its syntactic tag KP (case phrase). Its instrumental usage (Butt and King, 2004) was covered by introducing the INST semantic tag at the syntactic level as KP.INST. During the annotation evaluation, an additional POS tag POSTP.CMP (comparative postposition) was introduced to identify the polysemous behavior of *sE* case marker as a comparative postposition. This behavior can be identified easily in comparative situations like *dOsrON sE zyAdah*

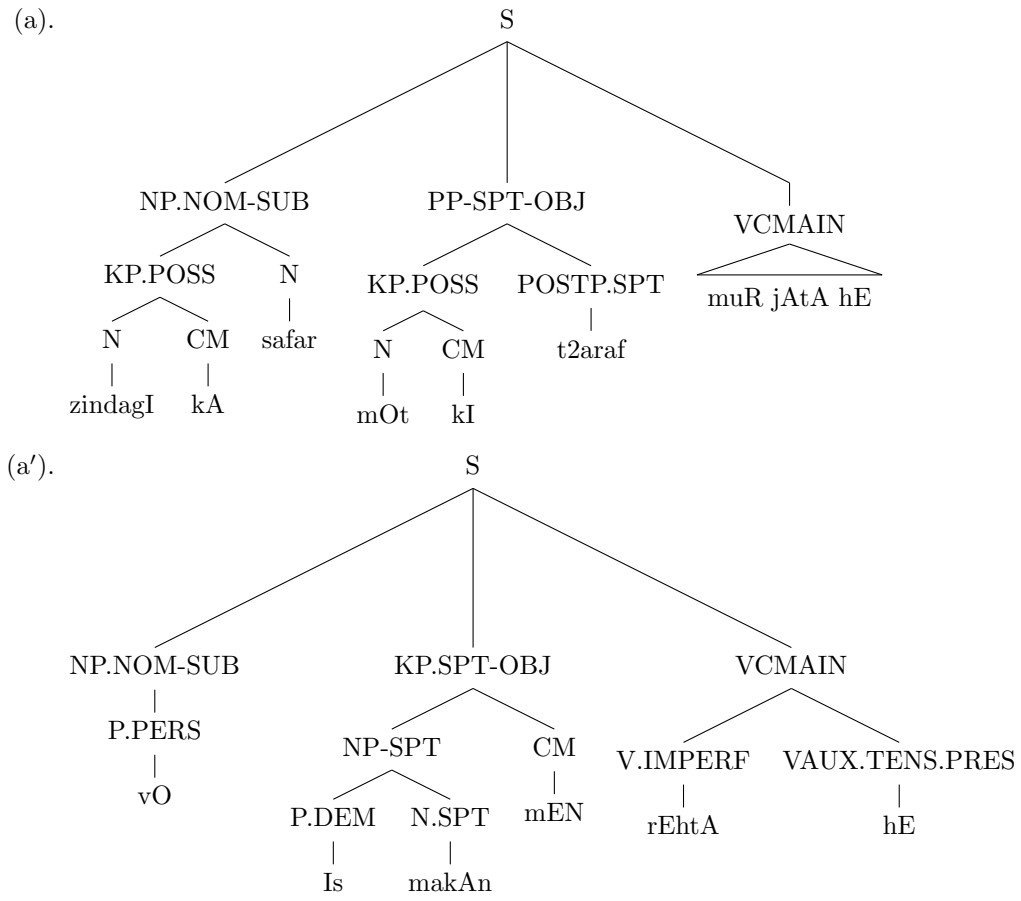


Figure 4.11: Initial annotation: a case of KP and PP as an object

'more than others'. Here the case marker *sE* is part of the comparative construction (Khan, 2009; Raza, 2011). The annotators' response on this phenomenon was quite good because its functional annotation was similar to SPT (spatial), TMP (temporal) and MNR (manner) semantic tags. The Figure 4.13(a, a′) shows the POS, syntactical and functional labeling of *sE* postposition for two phrases *dOsrON sE zyAdah* 'more than others' and *sab sE burA h2Al* 'worst situation' respectively. At the POS level, the *sE* 'than' is tagged as the POSTP.CMP, which means that the case marker *sE* is acting as a comparative postposition. As it is also acting as the head word in the both PP phrases, so its syntactic tag PP is added with the semantic tag CMP comes from the POS tag POSTP.CMP. Due

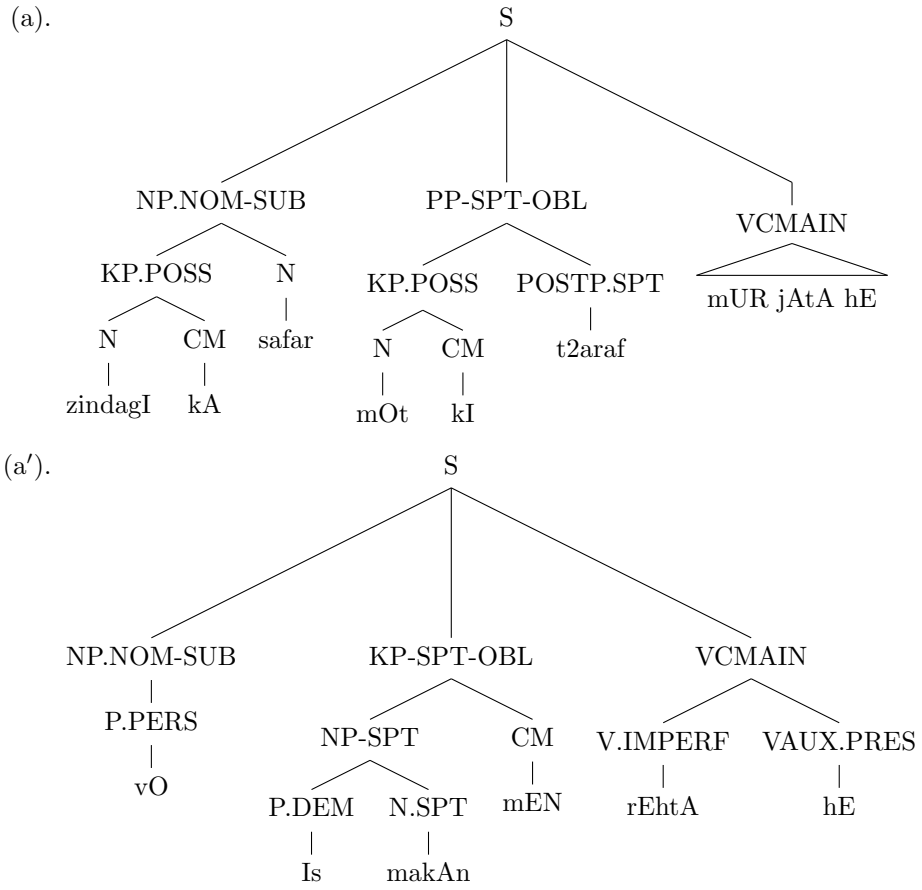


Figure 4.12: New F annotation: a case of KP and PP as an oblique (OBL)

to a good response of annotators, its POS and functional labeling was included in the updated tag sets depicted in Tables 3.3 and 3.6, respectively.

6. The initial version of the annotation scheme did not contain the MODF functional tag for the modifiers/adjuncts. It was adopted at the intermediate stage of the development and the guidelines which were given to the annotators during the training course contained this MODF tag. It was advised to assign the functional labels including the MODF to all the constituents, which are directly connected to root S level. The annotators performed this task sufficiently well and hence this tag was continued in the new version. An annotated example can be seen in Figure 4.14 for a sentence in 4.37.

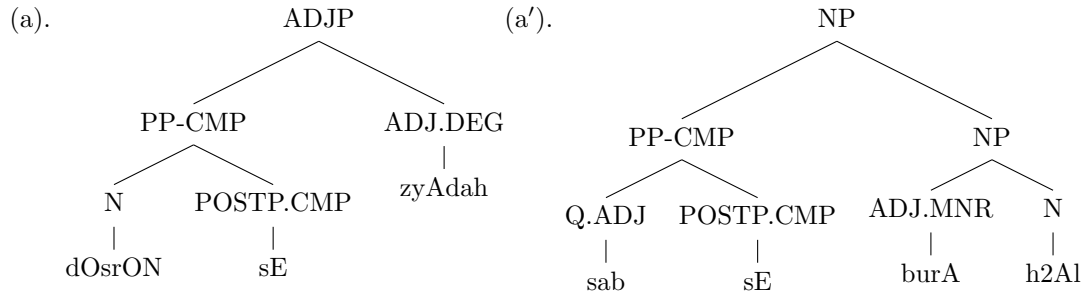


Figure 4.13: New F annotation: *sE* as POSTP.CMP

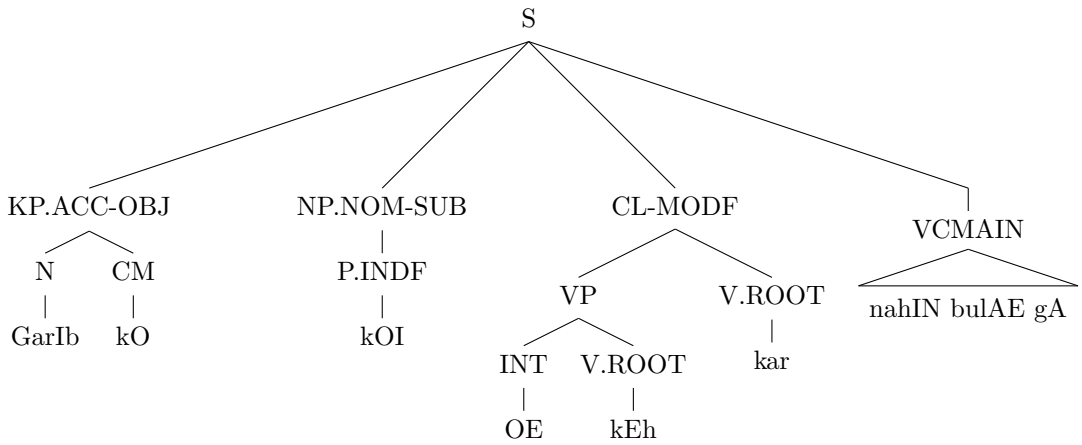


Figure 4.14: New F annotation: a case of -MODF

(4.37) *GarIb=kO* *kOI* *OE* *kEh* *kar* *nahIN* *bulAE* *gA*
poor=Acc nobody=Nom OE say.Root do.Root not call.Perf will.Fut
'Nobody will call the poor by saying "*OE*"'

The figure shows that all the constituents connected to S are labeled with the functional tags, which is essential according to the new version of annotation guidelines given in Chapter 2. Constituents like KP, PP, ADJP, ADVP, VP, CL, etc., can all act as a modifier in a sentence.

4.5 Revisions to the Annotation Scheme

This section presents an overview of general issues faced during the development of the URDU.KON-TB treebank and the training course of annotation evaluation. The issues resolved and the issues proposed with solutions are as follows:

1. The handling of the compound words in the case of manual annotation on the paper or in the text editors is not a complex task. The native annotators can identify these compound words easily. The compound words are separated by a space character '\u0020' or ASCII '32' in the text editors or the word processors; however, this insertion of a space character becomes a problem in automatic parsing because the space character is generally used as a word boundary.

   (4.38) *vO* *SAdI* *Sudah* *insAn* *hE*
   He=Nom marriage do.Perf+go.Perf human be.Pres
   'He is a married person'

   In sentence 4.38, *SAdI Sudah* (married) is a compound word, whose script form in Urdu is شادی شدہ. If we ignore the space between these words then the two words combine with each other due to the joiner ی at the end of the first word and ش in the beginning of the second word like this شادیشدہ, which is wrong orthographically. During the parsing this issue was removed by inserting a zero width non-joiner '\u200C' between the words of a compound word instead of a space character '\u0020' or ASCII '32'. The zero width non-joiner is an invisible character normally used to handle these kind of problems for different natural languages. The predefined functions in the programming languages normally consider a space character '\u0020' or ASCII '32' to distinguish the boundary of a word. By using zero width non-joiner in compound words, individual words are able to keep their identity independently in a single entity as a compound word. Further discussion on this issue is taken place in Chapter 5.

2. The subordinate clauses are represented by the SBAR in the syntactic annotation. However, in the updated version of the annotation, not all the subordinate clauses are annotated with the SBAR. The clauses containing the C.CORD (coordinating conjunction) are not tagged with the SBAR. They are treated like the constituents in a sentence as can be seen in the bracketed form of a sentence 4.39. This convention is also part of the ParGram[1] effort and examples can be seen on the

[1] http://pargram.b.uib.no/

XLE-WEB online interface.[1]

(4.39) *vO* *xUS* *hOtE* *tHE* *aOr* *ina2Am* *dEtE*
He=Nom happy become.Imperf be.Past and reward give.Imperf
tHE .
be.Past
'They had been happy and gave rewards'

```
( S
    ( NP.NOM-SUB-1
        ( P.PERS   vOh )
    )
    ( ADJP-PLINK
        ( ADJ   xUS )
    )
    ( VCMAIN
        ( V.COP.IMPERF   hOtE ) ( VAUX.PAST   tHE )
    )
    ( C.CORD   aOr )
    ( NP.NOM-SUB-L-1   * )
    ( NP.NOM-OBJ
        ( N   ina2Am )
    )
    ( VCMAIN
        ( V.IMPERF   dEtE ) ( VAUX.PAST   tHE )
    )
    ( M.S   . )
)
```

3. According to the argument structure theory by Butt and King (2004), a dative object (secondary or indirect object) KP.DAT-OBJ2 can exist along with the nominative object (direct object) NP.NOM-OBJ in a sentence, which had been adopted in the initial version of the annotation scheme. Later on, we proposed that a secondary or indirect nominative object NP.NOM-OBJ2 can also occur along with the direct nominative object NP.NOM-OBJ in a sentence as can be seen in the bracketed form of example sentence 4.40.

(4.40) *mua2ASrE=kA* *har* *fard* *axlAqI* *pAbandIyON=sE*
society.Fem=Poss every person.Masc moral obligations.Fem=Inst
mubarrA *AzAdI=kA* *mAh2Ol* *apnA* *bUnyAdI*
without independence.Fem=Poss environment.Masc own fundamental

[1] `http://iness.uib.no/xle-web/xle-web`

h2aq *gardAntA* *hE*
right.Masc consider.Imperf be.Pres
'Every person of society considers the environment of independence without moral obligations his own fundamental right'

```
( S
    ( NP.NOM-SUB
        ( KP.POSS
            ( N   mua2ASrE ) ( CM  kA )
        )
        ( NP
            ( Q.ADJ   har ) ( N   fard )
        )
    )
    ( NP.NOM-OBJ2
        ( PP
            ( KP-INST
                ( NP
                    ( ADJ.MNR   axlAqI ) ( N   pAbandIyON )
                )
                ( CM   sE )
            )
            ( POSTP   mubarrA )
        )
        ( NP
            ( KP.POSS
                ( N   AzAdI ) ( CM   kA )
            )
            ( N   mAh2Ol )
        )
    )
    ( NP.NOM-OBJ
        ( P.POSS.REF   apnA )
        ( NP
            ( ADJ   bUnyAdI ) ( N   h2aq )
        )
    )
    ( VCMAIN
        ( V.IMPERF   gardAntA ) ( VAUX.PRES   hE )
    )
    ( M.S   . )
)
```

In this sentence, *gardAntA* 'consider' is the predicate of the sentence, which has a subject *mua2ASrE=kA har fard* 'every person of society' annotated with NP.NOM-SUB. The question is that the subject considers what? The answer is *apnA bUnyAdI h2aq* 'his own fundamental right', which is acting as a direct nominative object in the sentence and annotated as NP.NOM-OBJ. The predicate with subject and direct object *mua2ASrE=kA har fard apnA bUnyAdI h2aq gardAntA hE* 'Every person of society considers his own fundamental right' is still not enough to complete the information in this sentence. This incomplete information is producing another question that every person of society considers his own fundamental right for whom? The answer of this question is given in the part *axlAqI pAbandIyON=sE mubarrA AzAdI=kA mAh2Ol* 'the environment of independence without moral obligations' of the sentence, which is the candidate of secondary or indirect nominal object and hence annotated as NP.NOM-OBJ2 in the bracketed sentence. Due to low frequency of this type of sentences as compared to other discussed in the beginning of this issue, by chance, it was not included in the sentences given to annotators during the course of annotation evaluation. However, this annotation in our scheme is the candidate for the future evaluation.

4. The addition of semantic labels like SPT, TMP, INST, and CMP with '-' at the syntactic level is performed in the presence of the semantic labels at the POS of the head word/dependent word as discussed in Issues 1 and 5 of Section 4.4.3. These semantic tags are passed up further to the KP level if the phrase contains a case marker CM. As the case markers have many functions (Butt and King, 2004; Khan, 2009; Mohanan, 1994) like instrumental, source expression, material, locative, temporal expression, etc., which can be inferred from the phrase in which these are used. This is the concept that we are already doing. To make it more clear, the main POS category of case markers CM can be divided into subcategories as CM.SPT (spatial/locative), CM.TMP (temporal), CM.INST (instrumental), CM.CMP (comparative), which will be more consistent with our existing annotation and can improve the readability too. Unfortunately at present, we are staying with a single tag CM to handle all the types of case markers in our annotation, which can be updated after the analysis and evaluation in future.

5. Similarly, in the updated version of the annotation scheme, different question phrases like ADJPQ (adjective phrase having a question), ADVPQ (adverb phrase having a question), KPQ (case phrase having a question), NPQ (noun phrase hav-

ing a question) and QWP (phrase having a question) are formed by looking at the context of the question word after applying some rules discussed in Issue 3 of Section 4.4.2. Due to this annotation style, sometimes a reader can be confused as to the different kinds of question phrases. To make it easy, the main POS tag for question word QW can be subcategorized into QW.ADJ (adjectival question word), QW.ADV (adverbial question word), QW.N (question word having a noun), etc., in question phrases to improve consistency and readability as well, however, it can not be updated until a complete analysis and evaluation.

4.6 Summary

This chapter reported on the annotation evaluation and subsequent revision of the annotation scheme. The investigation of related work has also been reported. The selection of Krippendorff's α coefficient was advocated. After explaining the preliminary work needed, the evaluation methods of respective SSP, SSS and F annotation are described. It has been found that the annotators have an agreement of 0.964, 0.817 and 0.806 in case of SSP, SSS and F annotations respectively. The issues faced during the training course of annotation evaluation and the development of the URDU.KON-TB treebank were discussed. The resolved issues and the issues still pending with or without proposals were also presented. In the process of resolving the issues, the tags which were changed or removed were also mentioned in their respective sections. After this exercise, it is concluded that the annotation evaluation process only guaranteed the reliability of annotation schemes as advocated by Reidsma and Carletta (2008). It does not guaranty that the annotation scheme is suitable for machine learning (ML). To resolve this problem, an annotated data is normally parsed by an automatic parser and evaluated accordingly. The development of an automatic parser for Urdu language is described in Chapter 5.

5

Urdu Parser

In this chapter, the development and evaluation of a treebank based parser for Urdu are presented. An existing Earley (Earley, 1970) parsing model is extended for the morphologically rich free order language Urdu. By considering the state-of-the-art parsing requirements, a context free grammar with sufficiently encoded information is extracted from a Urdu treebank described in Chapter 3. The extended parsing model and linguistically rich grammar give us state-of-the-art results for Urdu language. This Urdu parser (Abbas, 2014b) gives 87% of f-score and outperforms the multi-path shift-reduce parser for Urdu, a two stage constraint based dependency parser for Hindi and a simple Hindi dependency parser.

5.1 Background

State-of-the-art parsing systems are based on treebank grammars but unfortunately, for both constituency and dependency parsing, treebank based techniques are suffering in case of MRLs (morphologically rich languages) such as Czech (Collins et al., 1999), German (Dubey and Keller, 2003), Italian (Corazza et al., 2004), French (Arun and Keller, 2005), Modern Standard Arabic (Kulick et al., 2006), Modern Hebrew (Tsarfaty and Sima'an, 2007) and many other (Tsarfaty et al., 2010). The data driven parsing models used for dependency parsing are not guaranteed to observe all morphological variants of word form. Efficient parsing results for MRLs are hard to achieve without explicit encoding of linguistic information (Tsarfaty et al., 2013).

Getting state-of-the-art parsing results for a MRL is a challenge till to date. According to Tsarfaty et al. (2010, 2013), without proper handling of morphological entities in sentences, promising results for MRLs can not be achieved. To overcome this problem,

one approach is to embed solutions in a parser in form of morphological component for segmentation and a syntactic component for parsing. A parser generates a formal output by looking at the events in data after assuming independence between the events. Complex morphosyntactic interactions among events may impose constraints, which lead to explicit encoding of morphological information at the syntactic level. Assigning a correct morphological signature to each word in the presence of extreme data sparseness is almost impossible. Due to which this solution is suffering. However, on the other hand, the best broad coverage and robust parsers to date have grammars extracted from treebanks, which are a collection of syntactically annotated sentences by humans. The depth of information encoded in an annotation correlates with parsing performance (Tsarfaty et al., 2013). However, the languages lying in the category of MRLs are not a single homogenous class of languages due to their own cross lingual variations such as the extent of morphology, flexible ordering, syncretism, fine-grained and unambiguous morphological markers, etc.

The problem statement described above requires an explicit encoding of morphological information, which can be made if you have a treebank with sufficiently encoded information at POS, syntactic and semantic level of its annotation. A hierarchically annotated treebank for Urdu (Abbas, 2012) described in Chapter 3 was built to meet this requirement. A context free grammar (CFG) is extracted from the treebank computationally. The development procedure and the depth of encoded information in grammar is presented in Section 5.4. The grammar is then given to an extended dynamic programming parsing model which is known as the Earley algorithm (Earley, 1970). The extensions made to Earley's algorithm are given in Section 5.5. This algorithm is language independent and is capable to parse MRLs like CYK (Cocke-Younger-Kasami) as advocated by Tsarfaty et al. (2013) and Abbas et al. (2009). The reason that Earley's algorithm is adopted instead of CYK algorithm is that CYK takes grammar and produces parse trees both in Chomsky Normal Form (CNF) or binary trees, while the Earley algorithm takes grammar and produces parse trees both in any form, hence well suited to treebank based grammars. A CNF form can be converted into CFG but a model with less issues is adopted in the beginning because more issues during development are expected as discussed in Section 5.6. By applying a sufficiently rich grammar along with the extended parsing model, promising results are obtained and discussed in Section 5.7. Section 5.8 concludes the parsing development with future directions. Similarly, the related work of parser development in the domain of Urdu language is described in Section 5.2, which sets a path towards the construction of a state-of-the-art Urdu parser.

The traditional Earley parsing algorithm is normally taught in natural language processing (NLP) related courses at almost every educational institution in the world. It lies in the domain of chart parsers like CYK. No reduplication of effort and has a top down search with bottom up filtering. Left recursion is not a problem in this algorithm. It can handle null productions efficiently in contrast of CYK, which does not treat null productions due to CNF. It can solve an exponential problem in cubic time $O(n^3)$, an unambiguous grammar in quadratic time $O(n^2)$ and almost all left recursive grammars in linear time $O(n)$ and hence its time and space complexity is better than CYK. For n words in a sentence, it creates $n+1$ charts to provide a solution. It can handle almost every type of grammar and not only a CNF like the CYK algorithm. The Earley algorithm mainly has three procedures e.g. predictor, completer and scanner, which can be seen in Algorithm 5.1.

The algorithm takes words of a sentence and grammar productions as arguments. A dummy or a starting state is initiated in $chart[0]$ with the following production $\gamma \rightarrow \bullet S$ having dot symbol in the beginning of the right hand side (RHS). The outer loop at line 3 runs one more time than the length of a sentence. The inner loop at line 4 runs for all states in a respective chart. If the state is a complete state like $B \rightarrow \gamma\bullet$ having dot symbol at the end then the COMPLETER will find all the related productions like $A \rightarrow \alpha \bullet B\beta$ in $chart[j]$ and then it will add them in their updated forms (moving dot symbol forwarded) as $A \rightarrow \alpha B \bullet \beta$ in $chart[k]$ as can be seen in COMPLETER procedure. If the state is an incomplete state then the next category of the state would be checked. If the category of the state is found to be a terminal production then the SCANNER will simply match the candidate word of this production $A \rightarrow \alpha \bullet B$ with the current word $word[j]$ in a sentence and then it will add the updated production $B \rightarrow word[j]$ in the next chart $chart[j+1]$ as can be seen in SCANNER procedure. A state having incomplete status with a nonterminal calls the PREDICTOR procedure. For a production like this $A \rightarrow \alpha \bullet B$ in chart $chart[j]$, a PREDICTOR fetches all relevant productions like $B \rightarrow \gamma$ from $grammar$ rules and simply adds them as $B \rightarrow \bullet\gamma$ in the current chart $chart[j]$. In the last chart, the starting state comes again with its updated state as $\gamma \rightarrow S\bullet$, which means that a solution has been obtained and vice versa otherwise.

Algorithm 5.1 Earley Parsing Algorithm (Jurafsky and Martin, 2009)

```
1: function EARLEY-PARSE(words, grammar)
2:     ENQUEUE((γ → •S, 0), chart[0])
3:     for i ← 0, LENGTH(words) do
4:         for each state in chart[i] do
5:             if INCOMPLETE?(state) then
6:                 if NEXT-CAT(state) is a nonterminal then
7:                     PREDICTOR(state, i, grammar)              ▷ non-terminal
8:                 else
9:                     SCANNER(state, i)                         ▷ terminal
10:                end if
11:            else
12:                COMPLETER(state, i)
13:            end if
14:        end for
15:    end for
16:    return chart
17: end function
18: procedure PREDICTOR(((A → α • B, i), j, grammar))
19:    for each (B → γ) in GRAMMAR-RULES-FOR (B, grammar) do
20:        ADD-TO-SET ((B → •γ, j), chart[j])
21:    end for
22: end procedure
23: procedure SCANNER(((A → α • B, i), j))
24:    if B ⊂PARTS-OF-SPEECH(word[j]) then
25:        ADD-TO-SET ((B → word[j], i), chart[j + 1])
26:    end if
27: end procedure
28: procedure COMPLETER(((B → γ•, j), k))
29:    for each (A → α • Bβ, i) in chart[j] do
30:        ADD-TO-SET((A → αB • β, i), chart[k])
31:    end for
32: end procedure
```

5.2 Related Work

As for the related work of Urdu parser development is concerned, the first attempt was made in the ParGram[1] project by Butt and King (2007) using a LFG framework, then a two stage constraint based dependency parser for Hindi was introduced by Bharati et al. (2008). A NU-FAST treebank based grammar was used to parse the Urdu sentences using a built-in Prolog parser by Abbas et al. (2009). A dependency based parser was developed by Bharati et al. (2009a) for Hindi, which is a very close language to Urdu except the script writing style and the differences in the formal and informal versions. Later on, a dependency parser for Urdu was tried by Ali and Hussain (2010) using MaltParser (Nivre et al., 2007). Similarly, some experiments on dependency parsing for Urdu were performed by Bhat et al. (2012b) using MaltParser and MST (Minimum-Spanning Tree) (McDonald et al., 2005) parsers. In the same year, an Urdu probabilistic parser using shift-reduce algorithm was developed by Mukhtar et al. (2012b).

In the Urdu ParGram project (Butt and King, 2007), the XLE[2] parser is in use. The encoding of LFG grammar in XLE interface is not a simple task. Such a grammar can be encoded only by those persons who have expertise in theoretical linguistics as well. The team of the ParGram project has made a tremendous effort in this regard. This project of Urdu LFG grammar development is still in progress and the parser evaluation results are not available yet.

The two stage dependency parser for Hindi by Bharati et al. (2008) was trained on the Hindi treebank (Begum et al., 2008), which was annotated according to the Paninian grammatical model (Begum et al., 2008). The annotation scheme was designed on the basis of chunks, intra-chunks and karakas[3]. This scheme (Begum et al., 2008) of dependency structure (DS) is different from the annotation scheme (Abbas, 2012) of phrase structure (PS) and the hyper dependency structure (HDS) of the URDU.KON-TB treebank along with the different data sets used. As compared to phrase/constituent structure, the dependency structure lacks in information at non-terminal nodes (Bharati et al., 2008) and often the information at POS level. This information can also be provided at dependency annotation but people are stick to the standard norms. The Hindi treebank is rich in functional information as compared to morphological, POS and syntactical information. Due to differences in designs of the two stage dependency parser for Hindi and the Urdu parser, only the results are compared, which are presented

[1] http://ling.uni-konstanz.de/pages/home/pargram_urdu/

[2] http://www2.parc.com/isl/groups/nltt/xle/

[3] Karakas are the syntactico-semantic relations between the verbs and other related constituents in a sentence (Bharati et al., 1996)

in Section 5.7.

The Prolog parser used for evaluation of the NU-FAST treebank (Abbas et al., 2009) was in fact not the case of parser development, which means that a built-in utility in the inference engine of Prolog was used to parse the sentences. This utility can only be used if you have a definite clause grammar (DCG) or probabilistic definite clause grammar (PDCG). In this work, a CFG was extracted computationally through the NU-FAST treebank, then probabilities of grammar rules were counted. Through this process, a probabilistic context free grammar (PCFG) was obtained, which was converted into a PDCG format computationally. This PDCG was then given to the built-in utility of Prolog inference engine, which is basically a built-in Prolog parser to parse natural languages. In this work, a parser was not developed but a built-in parser was used, which concludes that the work is not an algorithmic development of parser that can be compared as explicitly mentioned in (Abbas et al., 2009).

A simple dependency parser for Hindi was developed by Bharati et al. (2009a). The parser used a grammar oriented approach, which was designed on the basis of Paninian grammatical model (Begum et al., 2008; Bharati et al., 1995) discussed earlier. The differences between the two stage dependency parser for Hindi (Bharati et al., 2008) and the Urdu parser presented earlier are also applied in the case of the simple dependency parser. Due to which only performance results can be compared but not the designs. The maximum results reported in labeled-attachment recall were 65.4% and 58.2% for chunks and intra-chunks, respectively. An average recall of karakas is calculated as 46.67%. A comparative study has been performed and presented in Section 5.7.

Ali and Hussain used the MaltParser with its default settings in Urdu dependency parser (Ali and Hussain, 2010). When somebody performs experiments with MaltParser with its default settings then such evaluation results are advised not to be compared according to MaltParser license.[1] The same exercise for parsing Hindi was performed by Agrawal et al. (2013), but it was clearly mentioned in the work that MaltParser was being used for error detection in the annotation of Hindi/Urdu treebank (HUTB).[2] Similarly, the Urdu sentences were parsed by Bhat et al. (2012b) using the same MaltParser. The experiments were performed to identify the parsing issues of Urdu and a development of parser was not claimed, neither it is the case of parser development. Moreover, the data-driven systems are highly criticized on a given set of annotated corpus because they are not able to observe all morphological variants of a word form from it (Tsarfaty et al., 2013).

[1] http://www.maltparser.org/

[2] http://faculty.washington.edu/fxia/treebank/

A multi-path shift-reduce parsing algorithm was proposed by Jiang et al. (2009) for Chinese. Later on, this algorithm was used for Urdu parsing by Mukhtar et al. (2012b). A probabilistic context free grammar (PCFG) developed by Mukhtar et al. (2011) was given to this parsing model depicted in Algorithm 5.2. It takes a part of speech tagged sentence as input. A *stack* and a *queue* are initialized with a null and a tagged word sequence respectively and then this initialization *ini* is stored in an array V. The total bunches of transitions are $2|x| - 1$ for which a *step* loop executes. A *state* loop runs for each state in V. This loop contains another *action* loop that decides about shift or reduce action to be taken on each state. After applying the action on a state, it is stored in a *next* variable which is then recorded in a buffer BUF. When the *state* loop ends, the BUF contains all constructed states in it. From which K best states are stored further in an array V. The *step* loop continues and the whole process from line 3 to 11 is repeated until it terminates. Finally, a tree is derived from best available states in V and displayed accordingly.

Algorithm 5.2 Multi-path shift-reduce parsing model (Jiang et al., 2009)

Input: POS-tagged word sequence x

1: $(ini.stack, ini.queue) \leftarrow (\emptyset, x)$
2: insert *ini* into **V**
3: **for** $step \leftarrow 1..2|x| - 1$ **do**
4: **for each** *state* in **V do**
5: **for each** *action* that can be applied to *state* **do**
6: *next* ←apply *action* to *state*
7: insert *next* into **BUF**
8: **end for**
9: **end for**
10: **V** ← **K** best states of **BUF**
11: **end for**

Output: the tree derived from the best state in **V**

A multi-path shift-reduce parser for Urdu has some limitations. It takes a tagged sentence as input and is not able to parse sentences without POS tagging. The stack used has a fixed memory size, which is not reliable and it can overflow during parsing of long sentences. A PCFG used in this parsing model is ambiguous (Mukhtar et al., 2012a). Both the fixed memory size and ambiguous grammar can resist the parsing of long sentences, that's why the parser could not parse sentences with length more than 10 words (Mukhtar et al., 2012b). In this work, the results were not evaluated

properly by using some measure e.g. PARSEVAL. A number of 74 sentences having length not more than 10 words were parsed successfully from 100 sentences, which were then quoted as a 74% of accuracy. The raw corpus used in the development of this parser is partially same as compared to the Urdu parser (Section 5.5). A comparative study made is detailed in Section 5.7.

By considering the modern parser requirements for MRLs by Tsarfaty et al. (2013) and the observations discussed in this section for Urdu and Hindi, it is a claim that the Earley based Urdu parser has the state-of-the-art evaluation results so far for both the languages. The morphological information encoded at POS, syntactic and semantic level is already discussed in Chapter 3, however a short overview in respect of parsing is discussed in Section 5.4. An Urdu parsing model is described in Section 5.5. Issues emerged during development and testing of the parser on the URDU.KON-TB treebank based grammar are addressed with proposed solutions. The respective algorithms and their discussion are given in Section 5.6. The results are evaluated using the PARSEVAL measure. A comparative evaluation with dependency parser (Bharati et al., 2009a) for Hindi and multi-path shift-reduce parser (Mukhtar et al., 2012b) for Urdu is discussed in Section 5.7. Possible future extensions and the conclusion are given in Section 5.8.

5.3 Motivation

Tsarfaty et al. (2010) discussed issues related to the encoding of morphological information and annotation schemes to be adopted for MRLs (Section 5.1). These issues highly advocate the need of treebanks annotated with sufficient morphological information. To fulfill this purpose, a treebank for Urdu with sufficient morphological information encoded at POS, syntactic and functional level is constructed (Abbas, 2012), see in Chapter 3. A statistical evaluation of the URDU.KON-TB treebank through annotated data of annotators was given in (Chapter 4), through which it has been concluded that the annotation scheme is reliable, but the answer of a question that the treebank is suitable for machine learning (ML) has not been found yet. To answer this question, an automated evaluation of the treebank through parser is needed. To achieve this objective, a parser is developed, provided with a grammar. This grammar is extracted from the URDU.KON-TB treebank discussed in Section 5.4. The parser can help the linguists analyze the Urdu sentences computationally and it can be useful in Urdu language processing (ULP) and ML domains. By using this parser, the size of the treebank can also be increased. This can be done after getting partial parsed trees

of unknown sentences. These partial parsed trees can be corrected and then imported into the URDU.KON-TB treebank. Using this methodology, one can speedily increase the size of treebank.

5.4 Preliminary Work

A treebank having PS (phrase structure) and HDS (hyper dependency structure) annotation with rich information is described in Chapter 3. This URDU.KON-TB treebank is used for the training of an Urdu parser discussed in Section 5.5. The URDU.KON-TB treebank has a semi-semantic POS (SSP) tag set, a semi-semantic syntactic (SSS) tag set and a functional (F) tag set. Morphological information in the labeling of the parser's lexicon is contained in the SSP tag set of the URDU.KON-TB treebank. A detailed discussion was already presented in Chapter 3, however, a short overview is presented here for ease of comprehension.

The SSP tag set hierarchy has 22 main tag categories which are divided into subcategories based on morphology and semantics. In Figure 5.1, an example of only a verb V is given. A dot '.' symbol is used for the representation of morphology and semantics at POS level. In Figure 5.1, the hierarchy of tag labels for verb V is divided into three levels of depth. The first level contains only one label to distinguish a verb V from other POS labels. The second level contains 11 subcategories of V to represent different morphological or functional forms e.g. V.COP (V as a copula verb), V.IMPERF (V has an imperfective form), V.INF (V has an infinitive form), etc. The third level contains further 25 subcategories to represent the morphological information in depth e.g. V.COP.IMPERF (copula verb has an imperfective form), V.COP.PERF (copula verb has a perfective form), V.COP.ROOT (copula verb has a ROOT form), V.COP.PAST (copula verb has a past tense), V.LIGHT.PAST (light verb has a past tense), etc. These types of combinations are also possible in case of an auxiliary verb as described in Chapter 3. This short discussion is about the idea of morphological and functional information encoded at POS level. This lexical information can be passed up to the syntactical level because the lexical items have some relationship with other lexical items in a sentence. We will explain it after presenting a short overview of the SSS and F tag set hierarchy.

The syntactic (SSS) and functional (F) tag sets contain 26 and 18 tag categories respectively. The SSS tag set hierarchy is divided into three more levels known as an hierarchical design. An example of a simple noun phrase (NP) is depicted in Figure 5.2. This is taken from the SSS tag set discussed in Chapter 3. A NP is divided into

V (Verb)
. COP (Copula verb)
. IMPERF (Imperfective copula verb)
. PERF (Perfective copula verb)
. ROOT (Root of copula verb)
. SUBTV (Subjunctive form of copula verb)
. PAST (Past tense of copula verb)
. PRES (Present tense of copula verb)
. IMPERF (Imperfective verb)
. REP (Repetition of Imperfective form of verb)
. INF (Infinitive form verb)
. LIGHT (Light verb)
. IMPERF (Imperfective form of light verb)
. INF (Infinitive form of light verb)
. PERF (Perfective form of light verb)
. PROG (Progressive form of light verb)
. ROOT (Root form of light verb)
. SUBTV (Subjunctive form of light verb)
. PAST (Past tense of light verb)
. PRES (Present tense of light verb)
. LIGHTV (Light verb with verb)
. IMPERF (Imperfective light verb with verb)
. INF (Infinitive light verb with verb)
. PERF (Perfective light verb with verb)
. ROOT (Root light verb with verb)
. SUBTV (Subjunctive light verb with verb)
. MOD (Modal verb)
. IMPERF (Imperfective modal verb)
. PERF (Perfective modal verb)
. SUBTV (Subjunctive modal verb)
. PERF (Perfective form of verb)
. REP (Repetition of perfective form of verb)
. ROOT (Root form of verb)
. REP (Repetition of root form of verb)
. SUBTV (Subjunctive form of verb)
. PAST (Past tense of verb)
. PRES (Present tense of verb)

Figure 5.1: A verb V example from the URDU.KON-TB treebank

NP (Noun Phrase)
.ACC (Accusative)
-OBJ (Direct Object)
-SUB (Subject)
.DAT (Dative)
-OBJ2 (Indirect Object)
-SUB (Subject)
-MNR (Manner)
.NOM (Nominative)
-MNR (Manner)
-OBJ (Direct Object)
-SUB (Subject)
-MODF (Modifier/Adjunct)
-OBJ (Direct Object)
-OBJ2 (Indirect Object)
-OBL (Oblique)
-PLINK (Predicate Link)
-SPT (Spatial)
-OBJ (Direct Object)
-PLINK (Predicate Link)
-SUB (Subject)
-SUB (Subject)
-TMP (Temporal)
-OBJ (Direct Object)
-OBL (Oblique)
-PLINK (Predicate Link)
-SUB (Subject)
.POSS (Possessive)
-SUB (Subject)
-MNR (Manner)
-MODF (Modifier/Adjunct)
-MODF (Modifier/Adjunct)
-SPT (Spatial)
-MODF (Modifier/Adjunct)
-OBJ (Direct Object)
-OBL (Oblique)
-TMP (Temporal)
-MODF (Modifier/Adjunct)

Figure 5.2: A noun phrase (NP) example from the URDU.KON-TB treebank

four levels of depth, as in the case of V. The first level is divided into 9 subcategories e.g. NP.ACC (accusative case of NP), NP.DAT (dative case of NP), NP-MNR (NP having a manner concept in it), NP.NOM (nominative case of NP), etc. The features are separated by dot '.' and dash '-'. In SSS annotation, '.' is used for representing syntactical features while the '-' is used for annotating semantic/functional features. The nine subcategories at the second level of NP annotation are further divided into 19 subcategories e.g. NP.ACC-OBJ (accusative case of NP acting as a direct object of a sentence), NP.ACC-SUB (accusative case of NP acting as a subject in a sentence), NP.DAT-OBJ2 (dative case of NP acting as an indirect object in a sentence), etc. At the last hierarchical level of the NP, NP.NOM-MNR, NP.NOM-SPT and NP.NOM-TMP are divided into two, three and four subcategories e.g. NP.NOM-MNR-OBJ (NP with nominative case having a concept of manner is acting as a direct object of the sentence), NP.NOM-SPT-OBJ (NP with nominative case having a spatial sense acting as a direct object in a sentence), NP.NOM-TMP-PLINK (NP with nominative case having a temporal sense acting as a predicate link of a sentence), etc.

(5.1) (a) میں اسے جنگل سے گزاروں گا

mEN	*usE*	*jangal*	*sE*	*guzArUN*	*gA*
P.PERS	P.PERS	N.SPT	CM	V.SUBTV	VAUX.FUTR

'I will pass him through the forest'

(b) میں اس کو جنگل سے گزاروں گا

mEN	*us*	*kO*	*jangal*	*sE*	*guzArUN*	*gA*
P.PERS	P.PERS	CM	N.SPT	CM	V.SUBTV	VAUX.FUTR

'I will pass him through the forest'

After presenting the SSP and SSS, we can now understand the passing of lexical information up to the syntactical level. Two sentences with respective trees are given in example 5.1 and Figure 5.3. Between these two sentences, the difference is only the presence of case marker *kO*. The passing up of lexical information at the POS to the syntactic level can be seen in the two different respective trees. In case of object marking as in Figure 5.3(a), the case marker (CM) *kO* with a personal pronoun (P.PERS) *us* 'him' is physically absent and the phrase contains only an inflectional form of personal pronoun as *usE* 'him' concluding a NP. This inflectional form has a hidden sense of CM *kO*, identifying an accusative case of direct object as NP.ACC-OBJ. On the other hand, the Figure 5.3(b) contains a CM preceded by a P.PERS. The physical existence of CM concludes a case phrase (KP) rather than a NP and disambiguating the trees as well.

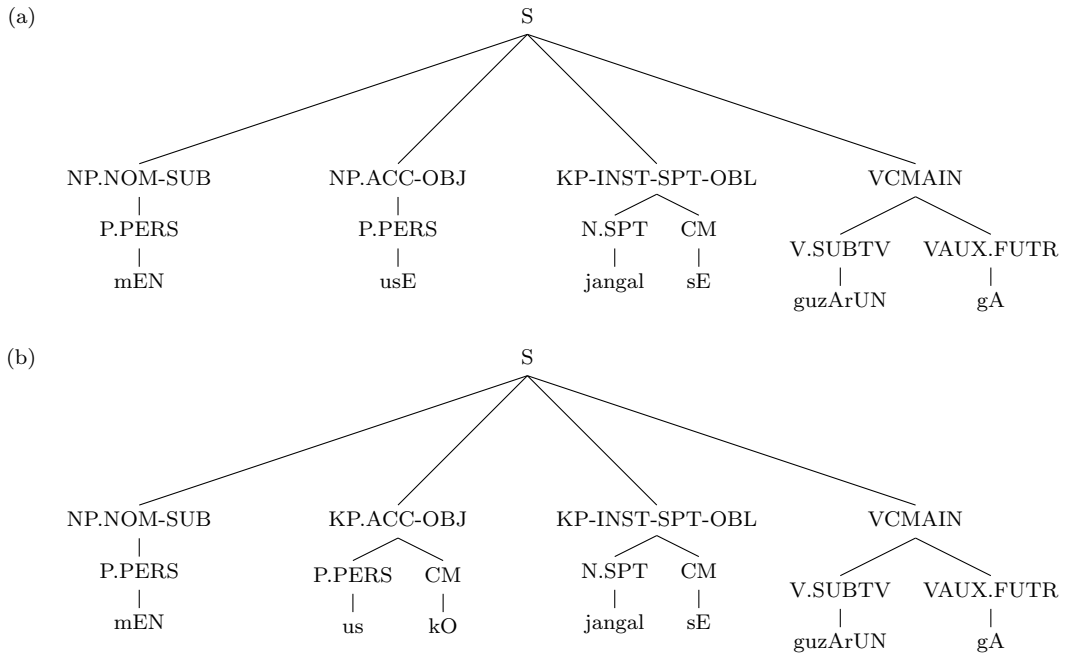


Figure 5.3: Passing up lexical information to the syntactical level

In the case of oblique OBL argument, the phrase *jangal/N.SPT sE/CM* 'through/from the forest' is annotated with KP due to instrumental case marker CM in both the trees. The instrumental INST semantic property of the CM is added up to the KP and similarly the word *jangal* 'forest' is a spatial noun whose spatial SPT semantic property is passed up to KP in order, which is acting as an oblique argument OBL in the sentence. The purpose of such information at morphological, POS, syntactical and functional level of annotation is not only to disambiguate the parse trees but it is essential to parse MRLs as concluded by Tsarfaty et al. (2013).

The URDU.KON-TB treebank is a manually annotated set of 1400 parsed sentences, which are then recorded in a text file on a computer in the form of 1400 bracketed sentences. Initial twenty bracketed-sentences from each hundred are separated in another text file, whose total 280 sentences are then used for development of a test suite discussed in Section 5.5. From the first residual text file with 1120 bracketed sentences, a context free grammar (CFG) is extracted using a stack based extraction module given in Algorithm 5.3. The text file contained bracketed sentences as displayed in Figure 5.6

separated with a '$' sign and is given as an input file to the algorithm. The algorithm used a *Stack* and a *StrArray* to process the productions between left and right parenthesis e.g. '(' and ')' of bracketed sentences. For example, If there is a production like (NP (N.SPT gHar)), then the algorithm pushes all strings one by one on to the *Stack* as (NP (N.SPT gHar until a first right parenthesis ')' after a token *gHar* 'house' is introduced. Without storing ')' on *Stack*, the already pushed strings are popped back until a left parenthesis '(' comes before a part of speech tag N.SPT and is stored respectively in an array *StrArray* in reverse order as gHar N.SPT without '('. At this point, a production/rule extraction has been completed in *StrArray*. The last element N.SPT of *StrArray* is copied back at the top of the *Stack* for further usage, because it is the part of next production to be extracted. The *StrArray* elements are recorded in an output file in its original order using lines 24 to 33 of algorithm concluding a production as N.SPT → gHar. After making *StrArray* empty, the reading of strings from input file remains continued. As the ')' appears, the popping back process from *Stack* has been started and the new production arrives in *StrArray* is N.SPT NP, which is then re-ordered as NP → N.SPT. This whole process continues until the end of file and finally an output file with a complete CFG productions of 1120 bracketed sentences is obtained. A CFG is depicted in Figure 5.4 for the example sentence in 5.2. The labeling as lexical (L) and non-lexical (NL) at the end of each production should not be considered as the part of output yet, however it will be explained in the forthcoming Section 5.5, which explains the design of the developed Urdu-parser.

(5.2)

vO	*xUS*	*hOtE*	*tHE*	*aOr*	*ina2Am*
he.Masc.Pl=Nom	happy	be.Cop.Imperf.Masc.Pl	be.Past.Masc.Pl	and	reward
dEtE	*tHE*	.			
give.Imperf.Masc.Pl	be.Past.Masc.Pl	full-stop			

'They had been happy and gave rewards.'

5.5 Urdu Parser

The output file obtained in Section 5.4 contained CFG productions extracted from 1120 bracketed sentences of the residual text file. The respective 280 unannotated (raw) sentences for building a test suite are separated from a second text file discussed in Section 5.4. The test suite is further divided into two halves representing test data and held out data resulting in 140 sentences in each half. The held out data was used in the development of the Urdu parser, while the test data was used for evaluation of results after the completion of the Urdu parser. The extracted CFG mentioned earlier

Algorithm 5.3 A CFG extraction algorithm

Input: A input and an empty output file

1: $(Sentence, Top, Counter) \leftarrow 0$
2: **Read**: InputString ▷ Read a string from an input file
3: **while** $InputString \neq Input.EOF()$ **do** ▷ Loop until end of file
4: **if** $InputString =$ \$ **then**
5: **Print**: $++Sentence$ ▷ Displaying sentence number on the screen
6: **Read**: InputString ▷ Read a string from an input file
7: **Write**: \n \n ▷ Writing two newlines in output file
8: $(Stack[0], StrArray[0]) \leftarrow \emptyset$ ▷ Initializing stack and array
9: $(Top, Counter) \leftarrow 0$ ▷ Initializing stack and array variables
10: **end if**
11: **if** $InputString \neq$ ")" **then**
12: $Stack[Top] \leftarrow InputString; Top++$ ▷ Filling stack with strings until ')' comes
13: **else** ▷ When ')' comes
14: $Top--$
15: **while** $Stack[Top] \neq$ "(" **do** ▷ Loop for popping back strings until '('
16: $StrArray[Counter] = Stack[Top]$ ▷ Storing a rule within '(' and ')'
17: $Stack[Top] = \emptyset$; $Counter++$; $Top--$
18: **end while**
19: $Counter--$
20: $Stack[Top] = StrArray[Counter]$ ▷ Storing rule LHS for further process
21: $Top++$ and $Check = Counter$
22: **while** $Counter \geq 0$ **do** ▷ Loop for writing CFG productions in output file
23: **if** $Counter = Check$ **then**
24: **Write**: $StrArray[Counter] \rightarrow$; $StrArray[Counter] = \emptyset$
25: **else**
26: **Write**: $StrArray[Counter] +$ ""; $StrArray[Counter] = \emptyset$
27: **end if**
28: $Counter--$
29: **end while**
30: **Write**: \n; $Counter = 0$ ▷ In output file
31: **end if**
32: **Read**: InputString ▷ Read a string from an input file
33: **end while**

Output: An output file having complete CFG productions for each sentence

```
P.PERS → vOh L
NP.NOM-SUB-1 → P.PERS NL
ADJ → xUS L
ADJP-PLINK → ADJ NL
V.COP.IMPERF → hOtE L
VAUX.PAST → tHE L
VCMAIN → V.COP.IMPERF VAUX.PAST NL
C.CORD → aOr L
NP.NOM-SUB-L-1 → * L
N → ina2Am L
NP.NOM-OBJ → N NL
V.IMPERF → dEtE L
VAUX.PAST → tHE L
VCMAIN → V.IMPERF VAUX.PAST NL
M.S → . L
S → NP.NOM-SUB-1 ADJP-PLINK VCMAIN C.CORD NP.NOM-SUB-L-1 NP.NOM-OBJ
     VCMAIN M.S NL
```

Figure 5.4: A CFG of the example sentence in 5.2

is processed by the Urdu parser to obtain a CFG database having unique productions from it. The grammar database has 6478 unique productions which include 828 S productions, 223 NP productions, 70 NP-SPT productions, 42 ADJP production, 352 KP productions with all morpho-syntactic and functional possibilities, etc. During this process, the labeling as L and NL at the end of each production is done. The productions having only lexical items at their right hand side are labelled as L and the productions which have non-lexical items on their right hand side are labelled as NL. The purpose of this labeling is to provide an already processed mechanism through which the parsing algorithm identify a production as L type or NL type speedily without checking it thoroughly. This identification helps the parser to select `PREDICTOR()`, `COMPLETER()` or `SCANNER()` for further process.

Without handling the issues discussed in Section 5.6, the Earley's algorithm is not able to provide state-of-the-art evaluation results for Urdu. These issues caused parsing discontinuities or parsing failures (the parser failed to parse next during processing of

a grammar production), due to which modifications or extensions were applied on the basic algorithm to remove these parsing anomalies. The extended version is depicted in Algorithm 5.4 as follows. In this algorithm, a given CFG grammar as a database has three fields/columns in the form of LHS (Left Hand Side of production), RHS (Right Hand Side of production) and Type (Type of production either L or NL), which can be inferred through matching with the CFG in Figure 5.4. After taking a sentence as input, various variables are initialized along with a `chart[]` array, which is an array of objects. The starting value of the chart is given as `ROOT @ S`. In place of bullet/dot symbol '•' used in Earley algorithm, here an '@' symbol is used because the bullet/dot symbol is extensively used in the hierarchal annotation of the URDU.KON-TB treebank, from which grammar productions are extracted. The working of the algorithm is similar to Earley's algorithm except modifications in `PREDICTOR(), COMPLETER()` and a `SCANNER()` along with the additional functions i.e. an `EDITOR()` for an automatic editing of discontinuous parses and a `BUILDER()` for building parse trees presented at the end of algorithm.

Before the execution of the internal loop, the `scannerFlag` is set to false, which will be dealt within scanner Algorithm 5.6, and an initial state of the chart is printed. Within the internal loop, the productions inserted in the respective charts are picked up one by one and are split with space to find the next processing non-terminal in `tempString` and its location `index`. If the `tempString` is found to be empty then the `COMPLETER()` will be executed to complete the status of the relevant production by moving forward the '@' symbol, which will be discussed in Section 5.6.7 along with its Algorithm 5.11. Otherwise, there are two possibilities, which are decided on the basis of further productions extracted from the grammar database into a record set `rs`. The first possibility of `PREDICTOR()` will be executed if the value of `rs` is true, which means that it contains further extracted productions and then the `PREDICTOR()` will add predicted productions into the current chart, whose further detail is presented in Section 5.6.1 with the Algorithm 5.5. Similarly, the second possibility of `SCANNER()` will be executed if `rs` is found to be empty and then `SCANNER()` will create a new chart and it will then add a relevant L type production into this chart from a previous chart as will be illustrated in Section 5.6.2 along with its Algorithm 5.6.

During processing of `COMPLETER(), PREDICTOR()` and `SCANNER()`, parsing discontinuities can happen. To check these types of phenomena, an `EDITOR()` will come into action and it will remove all previous irrelevant entries and charts causing discontinuity up to an optimal point of parsing discovered earlier. The detailed discussion is presented in Section 5.6.6 along with its Algorithm 5.10. After the end of the external loop, the

Algorithm 5.4 Urdu Parser

```
1: function URDU-PARSER(grammar)
2:     Input: Sentence                                          ▷ reading a sentence
3:     (id, fi, fj, fid) ← 0   ▷ initializing StateID, failed i, failed j, and failed StateID
4:     chart[0].add("id", "ROOT @ S", "0,0", " ", "Seed")       ▷ initializing chart
5:     for i ← 0 to LENGTH(sentence[]) do      ▷ Loop for each word in a sentence
6:         scannerFlag ← false, id ← 1
7:         Print: chart[i] → (StateId, Rule, @Position, BackPointer, Operation)
8:         for j ← 0 to ChartSize[i] do                         ▷ Loop for chart entries
9:             currentRule ← chart[i].getRule(j).split(" ")  ▷ splitting rule with space
10:            (tempString, index) ←(string-after-@, @Position) in currentRule
11:            if tempString = " " then                ▷ Completer Case if NULL string
12:                call COMPLETER()                        ▷ calling completer procedure
13:            else
14:                rs ← All grammar rules with LHS = tempString
15:                if rs.next() ≠ false then                ▷ checking rs is not empty
16:                    call PREDICTOR()                    ▷ calling predictor procedure
17:                else
18:                    call SCANNER()
19:                end if
20:            end if
21:            if scannerFlag=false & j+1=chartSize[i] & i ≠LENGTH(sentence[]) then
22:                call EDITOR()
23:            end if
24:        end for
25:    end for
26:    call BUILDER()
27: end function
```

parsed solutions are generated and stored in the form of charts with entries, but not in the form of parsed trees. To represent parsed solutions in the form of bracketed parsed trees, a `BUILDER()` function is executed. This constructs parsed trees of solutions by manipulating the back-pointers calculated in `COMPLETER()` function. The `BUILDER()` is able to display all parsed solutions of a given sentence as discussed in Section 5.6.4 along with its relevant Algorithm 5.8, then the Algorithm 5.4 for the Urdu parser is exited with a complete generation of charts and bracketed parsed trees. The discussion of this main Algorithm 5.4 is done in this section, while the algorithms of functions called by the main algorithm are discussed in detail in the respective subsections of Section 5.6.

Technologies used for the development of the Urdu parser includes OpenJDK-7[1] (Open source Java Development Kit 7), Apache Tomcat V 7.0[2] and MySQL[3] using Eclipse IDE[4] (Integrated Development Environment). The algorithms presented in Sections 5.5 and 5.6 provides a taste of source code written in Java environment.

5.6 Analysis and Evaluation

5.6.1 Eliminating Useless Predictions

Earley's Predictor() adds useless productions in charts which causes a wastage of time during processing, and increases the chance of misleading direction towards a wrong solution or discontinuous parse if these useless productions are selected by the Earley's Completer() or Earley's Scanner() for further processing. Suppose, the current token to be parsed in an input sentence is خان 'Khan', which is a proper noun and there is a NL type production NP → @ N.PROP N.PROP residing in the current chart of the parser. Here, NP is the noun phrase, N.PROP is the proper noun and the '@' symbol is the equivalent of the bullet/dot • symbol of the Earley algorithm. The '@' symbol before a non-terminal on the RHS of the production is the case of predictor and the non-extended `PREDICTOR()` adds all the available L type productions of N.PROP into the chart along with the relevant production N.PROP → @ خان from the grammar. Even the addition of irrelevant productions are not required, only the relevant production N.PROP → @ خان has to be added in the chart. This addition of irrelevant/useless productions is also true for other lexical items e.g. adjectives, personal pronouns, case

[1] `http://openjdk.java.net/projects/jdk7/`
[2] `http://tomcat.apache.org/index.html`
[3] `http://dev.mysql.com/`
[4] `https://www.eclipse.org/downloads/`

markers, etc. These useless additions cause a wastage of time and increase the chance of misleading direction towards a wrong solution or a discontinuous parse/state. To resolve this issue, the `PREDICTOR()` is extended to abandon the selection and insertion of useless productions from the grammar into the charts as given in Algorithm 5.5 and its execution is discussed next.

When the main parsing Algorithm 5.4 calls the `PREDICTOR()` then it checks the type of production first either as NL or L using an external `if` condition. For both types, it finds the existence of predicted production in current chart `chart[i]` using a `for` loop controlled by the size of the current chart `chartSize[i]`. This loop from line 3 to 8 is for NL and from line 18 to 23 is for L type of productions. The loop will break if the predicted production is located in `chart[i]` and then it will be the case when there is no need to add this production into the chart. However, if it becomes unable to locate the predicted production then the value of `k` becomes equal to `chartSize[i]` and if it happens, then a `repeat-until` loop is used to add all extracted predicted-productions into the `chart[i]`. In Earley's algorithm, this process is same for both NL and L types productions, which causes the addition of irrelevant productions. To make it more accurate, the `PREDICTOR()` is divided into two halves to deal with NL and L type productions separately. In dealing with L type of productions, the `PREDICTOR()` is introduced with another condition displayed at line 26 of Algorithm 5.5, which enforces the predictor to add only relevant productions into the chart. It matches the token at RHS of predicted-production `rs.getString(2)` with the current token `Sentence[i]` in a given input sentence. This condition forces the `PREDICTOR()` to introduce only the relevant productions in charts and eliminates the limited possibility of misleading direction towards a wrong solution or a discontinuous parse. After removal of irrelevant productions, the wastage-time factor is reduced to optimal linear-time $O(n)$, which at least is required for processing of n number of tokens in a given input sentence.

5.6.2 Extended Scanner

In Earley's parsing algorithm, the `scanner()` only matches the RHS of the L type production with the current token in a given sentence and may cause a wrong selection with an incorrect part of speech tag at the LHS of this L type production. For example, the verb ہے 'is' has different tags in the grammar. It can act as an auxiliary in a sentence with present tense e.g VAUX.PRES → ہے. It can behave as a copula verb e.g. V.COP.PRES → ہے and it can also act as a main verb e.g. V.PRES → ہے. This concept of having more than one tag is true for other lexical items as well. So, if this L type production VAUX.PRES → @ ہے is the right candidate in the current chart

Algorithm 5.5 Predictor

```
1: function PREDICTOR(rs, i, id, chart, chartSize)
2:     if rs.getString(3) = "NL" then                    ▷ Checking non-lexical production
3:         for k ←0 to chartSize[i]-1 do
4:             cRule ← chart[i].getRule(k).split(" ")
5:             if rs.getString(1) = cRule[0] & cRule[1] ="@" then
6:                 break                                 ▷ production exists in chart already
7:             end if
8:         end for
9:         if k = chartSize[i] then                      ▷ True, if production does not exist in chart
10:            repeat
11:                chart[i].add(id, rs.getString(1)+" @ "+rs.getString(2), i+","+i, " ", "Predictor")
12:                chart[i].print(id, Rule(id), @Position(id), BackPointer(id), Operation(id))
13:                chartSize[i] ← id = id + 1
14:            until !rs.next()
15:            rs.close()                                ▷ closing Result Set handle of grammar
16:        end if
17:    else                                              ▷ Case of lexical (L) productions
18:        for k ←0 to chartSize[i]-1 do
19:            cRule ← chart[i].getRule(k).split(" ")
20:            if rs.getString(1) = cRule[0] & cRule[1] ="@" then
21:                break                                 ▷ production exists in chart already
22:            end if
23:        end for
24:        if k = chartSize[i] then                      ▷ True, if production does not exist in chart
25:            repeat
26:                if rs.getString(2) = Sentence[i] then
27:                    chart[i].add(id, rs.getString(1)+" @ "+rs.getString(2), i+","+i, " ", "Predictor")
28:                    chart[i].print(id, Rule(id), @Position(id), BackPointer(id), Operation(id));
29:                    chartSize[i] ← id = id + 1
30:                end if
31:            until !rs.next()
32:            rs.close()                                ▷ closing Result Set handle of grammar
33:        end if
34:    end if
35: end function
```

then the `scanner()` of the Earley algorithm can select other available productions from the grammar due to a check on the RHS only. This can cause the wrong solution or a discontinuous/failed state in the chart during parsing. To remove this issue, the `scanner()` is extended in the same way as was done with the `PREDICTOR()` in Section 5.5. This solution presented in Algorithm 5.6 solves the issue partially at this level, but it is completed in Section 5.6.6.

Algorithm 5.6 Scanner

```
1: function SCANNER(rs, i, id, tempString, Sentence, scannerFlag, fi, fj, fid, chart, chartSize)
2:     rs ← All grammar productions with LHS = currentRule[index − 1] & RHS = tempString
3:     if rs.next() ≠ false then                          ▷ Record set rs is not empty
4:         if i+1 ≠ LENGTH(Sentence)+1 then
5:             if tempString = Sentence[i] then
6:                 if !scannerFlag then
7:                     chart[i + 1] ← add("0", rs.getString(1)+" "+rs.getString(2)+
                                " @",i+","+(i+1), " ", "Scanner" )
8:                     chartSize[i + 1] = 1                ▷ new chart is initialized
9:                     scannerFlag = true
10:                    rs.close()
11:                    fi = i, fj = j, fid = id
12:                else
13:                    rs.close()
14:                end if
15:            else
16:                rs.close()
17:            end if
18:        else
19:            Print: Input Sentence ends at i
20:        end if
21:    else
22:        Print: Record set rs is empty
23:    end if
24: end function
```

When the `SCANNER()` is called, it extracts all relevant L type productions from grammar after matching their LHS and RHS with `currentRule[index - 1]` and `tempString` of current L type production in a chart, respectively. It adds the correct L type produc-

tion in a new chart `chart[i + 1]` after checking three conditions. At first, it checks that the chart number is not exceeding the length of the sentence, as the total number of charts to be produced should be equal to `LENGTH(Sentence)+1`. Secondly, it checks that the token `tempString` in the current processing L type production is equal to the current token `Sentence[i]` in a given sentence. After that if the `scannerFlag` is false, then the new entry of matched L type production is added into the new chart. During this process, the `scannerFlag` is set to true value along with a record of some variables `fi, fj, fid`, which will be used in `EDITOR()` presented in Algorithm 5.10. By introducing this modification, the possibility of wrong selection of production from the grammar is abandoned and the exactly matched L type production extracted from the grammar is added to the next chart accordingly. An issue left in this solution is that the Earley's `PREDICTOR` can add all the L type productions mentioned in Section 5.5 into the current chart for processing of `scanner()`. In this situation, the `scanner()` will not know which one is the true candidate production for parsing. This issue is addressed and resolved in Section 5.6.6.

5.6.3 Back-Pointers Calculation

Earley parsing algorithm is a generator or a recognizer and hence cannot produce parse trees or bracketed trees. The output of the Earley algorithm is the N+1 number of charts where N is the number of words in a given sentence. To produce parse trees or bracketed trees from the output of the Earley algorithm, an idea of back-pointers was discussed by Earley (1968) in his doctoral thesis. The same idea is implemented to produce parse trees in our Urdu parser. To understand the calculation of back-pointers, a sentence given in example 5.3 is parsed from the Urdu parser. The charts generated through it are depicted in Figure 5.5. Only relevant states are displayed as can be inferred from the non-sequential values of STATEID column. The column DOT-POSITION is basically the position of '@' in productions.

(5.3) ان کا ذکر بھی یہاں ضروری ہے

un	*kA*	*zikr*	*bHI*	*yahAN*
their/P.PERS	of/CM	reference/N	also/PT.INTF	here/ADV.SPT
zarUrI	*hE*			
essential/ADJ.MNR	is/V.COP.PRES			

'Their reference is also essential here'

CHART(0)				
STATEID	RULE	DOT-POSITON	BACK-POINTER	OPERATION
0	ROOT @ S	0,0		Seed
2	S @ NP.NOM-SUB ADVP-SPT-MODF ADJP-MNR-PLINK VCMAIN M.S	0,0		Predictor
52	NP.NOM-SUB @ KP.POSS N PT.INTF	0,0		Predictor
113	KP.POSS @ P.PERS CM	0,0		Predictor
148	P.PERS @ ان	0,0		Predictor
CHART(1)=ان				
STATEID	RULE	DOT-POSITON	BACK-POINTER	OPERATION
0	P.PERS ان @	0,1		Scanner
3	KP.POSS P.PERS @ CM	0,1	1-0	Completer
4	CM @ کا	1,1		Predictor
CHART(2)=کا				
STATEID	RULE	DOT-POSITON	BACK-POINTER	OPERATION
0	CM کا @	1,2		Scanner
2	KP.POSS P.PERS CM @	0,2	1-0 2-0	Completer
9	NP.NOM-SUB KP.POSS @ N PT.INTF	0,2	2-2	Completer
22	N @ ذکر	2,2		Predictor
CHART(3)=ذکر				
STATEID	RULE	DOT-POSITON	BACK-POINTER	OPERATION
0	N ذکر @	2,3		Scanner
1	NP.NOM-SUB KP.POSS N @ PT.INTF	0,3	2-2 3-0	Completer
12	PT.INTF @ بھی	3,3		Predictor
CHART(4)=بھی				
STATEID	RULE	DOT-POSITON	BACK-POINTER	OPERATION
0	PT.INTF بھی @	3,4		Scanner
1	NP.NOM-SUB KP.POSS N PT.INTF @	0,4	2-2 3-0 4-0	Completer
2	S NP.NOM-SUB @ ADVP-SPT-MODF ADJP-MNR-PLINK VCMAIN M.S	0,4	4-1	Completer
17	ADVP-SPT-MODF @ ADV.SPT	4,4		Predictor
74	ADV.SPT @ یہاں	4,4		Predictor
CHART(5)=یہاں				
STATEID	RULE	DOT-POSITON	BACK-POINTER	OPERATION
0	ADV.SPT یہاں @	4,5		Scanner
1	ADVP-SPT-MODF ADV.SPT @	4,5	5-0	Completer
2	S NP.NOM-SUB ADVP-SPT-MODF @ ADJP-MNR-PLINK VCMAIN M.S	0,5	4-1 5-1	Completer
3	ADJP-MNR-PLINK @ ADJ.MNR	5,5		Predictor
4	ADJ.MNR @ ضروری	5,5		Predictor
CHART(6)=ضروری				
STATEID	RULE	DOT-POSITON	BACK-POINTER	OPERATION
0	ADJ.MNR ضروری @	5,6		Scanner
1	ADJP-MNR-PLINK ADJ.MNR @	5,6	6-0	Completer
2	S NP.NOM-SUB ADVP-SPT-MODF ADJP-MNR-PLINK @ VCMAIN M.S	0,6	4-1 5-1 6-1	Completer
12	VCMAIN @ V.COP.PRES	6,6		Predictor
42	V.COP.PRES @ ہے	6,6		Predictor
CHART(7)=ہے				
STATEID	RULE	DOT-POSITON	BACK-POINTER	OPERATION
0	V.COP.PRES ہے @	6,7		Scanner
1	VCMAIN V.COP.PRES @	6,7	7-0	Completer
2	S NP.NOM-SUB ADVP-SPT-MODF ADJP-MNR-PLINK VCMAIN @ M.S	0,7	4-1 5-1 6-1 7-1	Completer
3	M.S @ -	7,7		Predictor
CHART(8)=-				
STATEID	RULE	DOT-POSITON	BACK-POINTER	OPERATION
0	M.S - @	7,8		Scanner
1	S NP.NOM-SUB ADVP-SPT-MODF ADJP-MNR-PLINK VCMAIN M.S @	0,8	4-1 5-1 6-1 7-1 8-0	Completer
2	ROOT S @	0,8	8-1	Completer

Figure 5.5: A back-pointer calculation example of the Urdu parser

The COMPLETER() Algorithm 5.11 calls the BACKPOINTER() Algorithm 5.7 to calculate the value of back-pointers. For example, during processing of STATEID 3 in chart 1 of Figure 5.5, the COMPLETER() calls the BACKPOINTER() with string type arguments previousRule and dummy@Position as KP.POSS→P.PERS @ CM and "0,1" respectively. The final indexed value of '@' in variable tempIndex after subtraction obtained through the previousRule is 1. The string value P.PERS at this index is then stored in a variable NT. The value of k becomes 0 at this stage in the algorithm. The external loop is used to traverse the charts one by one from the recent state of the chart. After checking the condition of tempIndex at line 8, another loop for traversal of chart entries is executed. At this point, a production P.PERS→ان @ at state 0 of chart 1 is stored finally in the cRule. The index of '@' here in this production is stored in the tIndex as 2. A condition presented next becomes true and the value of the backPointer is calculated as "1-0". The value in the DOT-POSITION column of this state is split with ',' and stored in a temporary variable dummy@P. Before breaking of internal loop, the values in l, tempIndex and NT are updated with 1, 0 and KP.POSS respectively. Then the control goes to the external loop in continuity with l=1, but now the condition of tempIndex becomes false and finally the external loop breaks. The calculated "1-0" value of backPointer is then displayed by the COMPLETER() in the relevant state of the chart. The rest of the back-pointers displayed in Figure 5.5 are calculated in the same way. These back-pointers are further used in building of bracketed parse-trees which will be discussed in Algorithm 5.8.

5.6.4 Building Bracketed Parse Trees

All the back-pointers are calculated and stored in each relevant state of the chart as presented in Section 5.6.3. When all the states in the respective charts are evaluated and the external loop in Algorithm 5.4 is exited then another function BUILDER() depicted in Algorithm 5.8 is called. After displaying all chart entries as shown in Figure 5.5, the possible bracketed parse trees are evaluated and displayed by the BUILDER() function. Both the BUILDER() and the BACKPOINTER() contributes to shift our Algorithm 5.4 from a generator to a parser in contrast of the Earley's algorithm.

After displaying the chart entries in Figure 5.5 for the sentence given in example 5.3, the BUILDER() sets the num and chartN to zero and the length of sentence, respectively. A loop is used to calculate the location of all S solution productions in the last chart and stored them in a string list bp[]. After evaluation, it has only one solution production with "8-1" string value in bp[] for the example sentence. A bracketed-tree class type array of objects tree[] is initialized with the number of solutions found at

Algorithm 5.7 Back Pointer

```
1: function BACKPOINTER(previousRule, dummy@Position, i, chartSize, chart)
2:     backPointer ← ""
3:     tempIndex ← previousRule.indexOf("@")                ▷ locating index of '@'
4:     tempIndex ← tempIndex-1                              ▷ subtracting index
5:     NT ← previousRule.get(tempIndex)                     ▷ saving non-terminal before '@'
6:     k ← dummy@Position[0]   ▷ initial '@' position of previous relevant chart rule
7:     for l ← i to k step -1 do                            ▷ loop for backward backpointers
8:         if tempIndex > 0 then
9:             for m ← 0 to chartSize[l]-1 do ▷ loop for locating relevant chart entries
10:                pString ← chart[l].getRule(m).split(" ")
11:                cRule.add(pString[])                     ▷ store pString in cRule
12:                tIndex ← cRule.indexOf("@")              ▷ getting index of '@'
13:                if (NT = cRule[0]) & (tIndex+1 = SIZE(cRule)) then
                   ▷ matching current non-terminal with non-terminals in relevant rules
14:                    backPointer ← (l + "-" + chart[l].getStateId(m) +" "+backPointer)
                                                            ▷ calculating backpointer
15:                    dummy@P = chart[l].get@Position(m).split(",")   ▷ getting '@' position
16:                    l ← dummy@P[0]                       ▷ updating loop counter l
17:                    l ← l + 1
18:                    tempIndex ← tempIndex-1
19:                    NT ← previousRule[tempIndex]
20:                    break
21:                else
22:                    cRule.clear()
23:                end if
24:            end for
25:        else
26:            break
27:        end if
28:    end for
29: end function
```

Algorithm 5.8 Builder

```
1: function BUILDER(Sentence[], chartSize[], chart[])
2:     num=0, chartN = LENGTH(Sentence[])
3:     for count ← chartSize[LENGTH(Sentence[])-1 to 0 step -1 do
4:         dummystr ←"S" and rule ← chart[chartN].getRule(count).split(" ")
5:         if rule[0] = dummystr then
6:             num = num + 1
7:             bp.add(chartN+"-"+chart[chartN].getStateId(count))   ▷ collecting relevant back-pointers
8:         end if
9:     end for
10:    tree[] ← new BTree[num]          ▷ initialized num number of bracketed-tree type tree[] objects
11:    for i ← 0 to SIZE(bp)-1 do
12:        tree[i].build(bp.get(i), chart)                          ▷ building tree with pointers
13:    end for
14:    for i ←0 to SIZE(bp)-1 do
15:        tree[i].prepare(chart)              ▷ preparing bracketed tree for presentation
16:    end for
17:    for i ←0 to SIZE(bp)-1 do                          ▷ loop for displaying all parsed trees
18:        bracketedSentenceLength ← tree[i].getSize() and left ← 0
19:        if bracketedSentenceLength > 0 then
20:            Print : Bracketed Parse Tree "+(i+1)+" of "+SIZE(bp)+"
21:            for j ←0 to bracketedSentenceLength-1 do
22:                if tree[i].getString(j) = "(" then
23:                    left = left + 1 and Print :  newline
24:                    for tab ←0 to left-1 do
25:                        Print : eight spaces
26:                    end for
27:                    Print : tree[i].getString(j)
28:                else if tree[i].getString(j) = ")" then
29:                    left = left − 1 and Print : tree[i].getString(j)
30:                else
31:                    Print : space+tree[i].getString(j)+space
32:                end if
33:            end for
34:        end if
35:    end for
36: end function
```

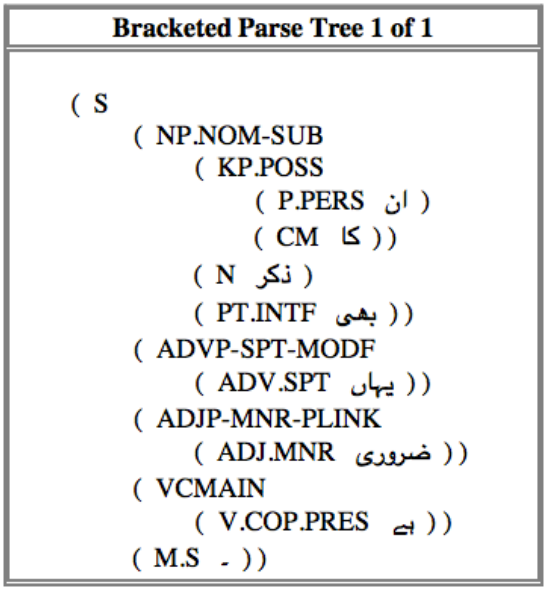


Figure 5.6: An output of the `BUILDER()` method

line 10 of the algorithm. This `tree` object is then called a user-defined method named `build(bp.get(i), chart)` with two arguments in another loop. This method builds an unformatted intermediate bracketed-parse-tree with interlinked back-pointers from chart states and reveals the leaf nodes only as (8-1 (4-1 (2-2 (1-0 ان) (2-0 کا)) (3-0 ذکر) (4-0 بھی)) (5-1 (5-0 یہاں)) (6-1 (6-0 ضروری)) (7-1 (7-0 ہے)) (8-0 .)). This intermediate parse tree can be understood well by looking at the given back-pointers in the respective chart states.

A third independent loop starts and prepares the intermediate parse tree into a complete unformatted parse tree using a statement given as `tree[i].prepare(chart)`. When this loop is exited then the parse tree has obtained a form as (S (NP.NOM-SUB (KP.POSS (P.PERS ان) (CM کا)) (N ذکر) (PT.INTF بھی)) (ADVP-SPT-MODF (ADV.SPT یہاں)) (ADJP-MNR-PLINK (ADJ.MNR ضروری)) (VCMAIN (V.COP.PRES ہے)) (M.S .)). So, this `prepare()` method only replaces the back-pointer with the LHS of relevant productions. At this point, the only thing left is the formatting of this parse tree, which is done from line 17 to 35 in Algorithm 5.8 and the parse tree gets a final form as depicted in Figure 5.6.

5.6.5 Empty Productions

Empty productions are divided into two categories. The first one is related to diacritic productions e.g. DIA → * and the second one is related to non-diacritic productions e.g. P.PERS → *. The POS tags DIA and P.PERS are used to represent diacritics and personal pronouns respectively, where the '*' is used to represent absence of these categories. The diacritic is a mark or a sign that is placed over, under, or through a letter of a word in some languages to describe that the letter of a word should be pronounced in a particular way or the word with a diacritic at the last letter has some sort of connection to the following word like compound words in Urdu which will be discussed next. As tags to represent these empty productions exist in our grammar, there is a need to deal it with. It can cause discontinuity during parsing because the lexical item may or may not be present for both the categories in a given sentence. A solution was proposed for compilers by Aho et al. (2007), Appel and Palsberg (2007) and Leblanc and Fischer (1988). A similar solution was coded for the Earley algorithm by Aycock and Horspool (2002), which has been adopted. The solution is to process the empty productions implicitly in the algorithm by moving the '@' symbol ahead. This solution causes frequent discontinuity during parsing and its property of self decision at irrelevant places makes the things more worse. It has been found that it does not work well for Urdu.

Usually, diacritics are not present in modern Urdu writing, but the older literature makes extensive use of diacritics e.g. a compound word آبِ حَیَات *AbE h2ayAt* 'The water of life' with a diacritic at the last letter of the first word *Ab* 'water' making it a possessive entity of the following word *h2ayAt* 'life', a single word تقرِیباً *taqrIban* 'almost' with a diacritic at the end of the last letter allowing this word to be pronounced as *taqrIba-n* with 'n' sound and not as *taqrIba*. The lexicon can be viewed on the website designed by the Center for Language Engineering[1] under the rights protected by the Ministry of Information and Technology, Pakistan.[2] The diacritic used in first example under the last letter بِ *b* of the first word is called *zErE-Iz3Afat* (a diacritic for addition), which is connecting the first word with the following word and thus making a compound word is still in use in modern Urdu writing. The diacritic above the last letter اً *a* in second example is called *tanwin* (a diacritic for final post-nasalization), which is different from the first example and related to pronunciation of the letter. There are also other diacritics in use as well e.g., *zEr, zabar, pEsh, taSdId*, etc. In short, the problem of the modern Urdu text is this that diacritics may arise in the text

[1] http://www.cle.net.pk/oud/

[2] `http://www.pakistan.gov.pk/`

or not.

During parsing, a compound word can come with the diacritic *zErE-Iz3Afat* in a given sentence e.g. شہرِمکہ *Sehr makkah* 'The city of Makkah'. This example has two words شہر *Sehr* 'city' and the مکہ *makkah* 'Makkah', but the diacritic *E* is absent at the end of the first word *Sehr*. In such cases, its presence is by default understood by the native speakers. The production exists in the grammar to handle this compound word is NP-SPT → @ N.SPT DIA N.PROP.SPT. When the first word *Sehr*/N.SPT is processed by the `SCANNER()` then the '@' is moved ahead and the production becomes NP-SPT → N.SPT @ DIA N.PROP.SPT. The DIA is the candidate to be executed next to represent the absence of diacritic at the end of first word *Sehr*. The `PREDICTOR()` deals this DIA empty production implicitly by moving the '@' ahead and adds the updated production NP-SPT → N.SPT DIA @ N.PROP.SPT in the same chart. Similarly, the second word *makkah*/N.PROP.SPT is processed by the `SCANNER()` and the production final state becomes like NP-SPT → N.SPT DIA N.PROP.SPT @. The problem with this solution by Aycock and Horspool (2002) is that it performs the implicit transaction silently for both the compound and non-compound words as observed and is unable to differentiate between these type of words. For example, if there is a case of compound word as discussed then the solution is perfect, but in the case of non-compound words if two independent words گھر *gHar* 'The house' and مکہ *makkah* 'Makkah' appear in a same position like discussed in compound words e.g. گھر مکہ میں ہے *gHar makkah mEN hE* 'The house is in Makkah', then this solution can not identify the context and it applies the transaction in the same way due to the same POS tagging of the *gHar* and the *Sehr*.

The diacritics are the part of words in Urdu. In the URDU.KON-TB treebank annotation, no tag was used for the presence of diacritics. It is a wonder, why the DIA tag is used to represent absence only. After facing a problem discussed earlier, the production of DIA was removed from a sample of the grammar and achieved quite good results. It is a tested proposal that this production of DIA should be removed to achieve good parsing results on this treebank.

The absence of an argument or an optional lexical item in the sentence is the case of non-diacritic empty production. In this case, the discontinuity rate is less than the other discussed earlier. It may be due to independent constituency structure of this category. For example, if a subject with nominative case is missing in a given sentence then the grammar has productions as NP.NOM-SUB → P.PERS and P.PERS → *. Both productions have the singletons (one terminal/non-terminal) on their RHS. This property of singletons is also existed for the other empty productions in this category.

The constituent structure is different from the category of diacritic productions. It is not sure but it may be the singletons on the RHS, which are preventing this category not to show immaturity during the parsing.

Algorithm 5.9 Empty Productions

```
1: if rs.getString(2) = "*" then
2:     chart[i].add(id, rs.getString(1)+" "+rs.getString(2)+" @", i+","+i, " ", "Predictor"
3: else
4:     chart[i].add(id, rs.getString(1)+" @ "+rs.getString(2), i+","+i, " ", "Predictor"
5: end if
```

Due to high frequency of the DIA production in the URDU.KON-TB treebank, the proposed solution of Aycock and Horspool (2002) was implemented in `PREDICTOR()` by replacing the lines 11 and 27 with the Algorithm 5.9 but the results found were not promising. So, an explicit method to represent empty productions has been adopted. A '*' is usually typed in the given sentence to represent absence of diacritics, arguments, lexical items, etc. Due to this explicit approach, the Urdu parser is jelling with the grammar without any issue related to empty productions.

5.6.6 Lexical Dynamic Behavior

The issue is related to the words which are homonyms, homographs, homophones, heteronyms and polysemes. A strict definition is considered to these attributes, that means at least the words have the same spelling. It is considered because the parser matches the tokens with the grammar lexicon or with the current word in a given sentence. The case of homonym words in a strict sense is going to be discussed here and the same concept is applicable on other attributes as well. There was an error in the Earley's algorithm, which has been removed in the Urdu parser. It was observed that when the Earley `Predictor` introduced the right candidate word along with some of its homonyms in the current chart and the Earley `Scanner` started its search for the right candidate word to be processed from the beginning of the current chart, then if homonyms of the right candidate word were present with different POS tags at the earlier positions in the current chart then the `Scanner` picked it up for processing and added it into the next chart after matching with the current token in the given input sentence. As this homonym had a different POS tag as compared to the right candidate word, so consequently the `Completer` and the `Predictor` processed further productions and finally the parser got into a wrong solution or a discontinuous state. This issue is explained with an example as follows.

For example, the word کی *kI* is a homonym in Urdu. It can behave in two ways e.g. a possessive case marker and a verb. Being a possessive case marker, it contains a possessive meaning 'of' in 5.4. On the other hand, it contains a meaning of 'did' in 5.5 as a verb.

(5.4) جولیا کی کتاب

jUlIA=kI	*kitAb*
Julia.Fem.Sg=Poss	book.Fem.Sg

'The book of Julia'

(5.5) اس نے ایک بات کی ہے

us=nE	*Ek*	*bAt*	*kI*	*hE*
he.Sg=Erg	a	talk.Fem.Sg	do.Perf.Sg	be.Pres.Sg

'He did a talk'

In the grammar of the Urdu parser, this word *kI* has different POS tags, as a case marker CM, a perfective verb V.PERF and a perfective light verb V.LIGHT.PERF. Suppose the word *kI* actually comes as a V.PERF at the end of a given sentence. For its processing, the `Scanner` can pick up the wrong choice with the CM or the V.LIGHT.PERF if these choices are available in the current chart at earlier positions as compared to the right choice. Due to this wrong selection, the relevant productions of the verb V.PERF will not be completed in the next chart and the parser will go with the wrong solution or the discontinuous state. To resolve this issue, it is needed to delete all the failed charts and continue the parsing process with the other choice. To address this issue, the `Scanner` of the Earley algorithm is modified in the Urdu parser as depicted in Algorithm 5.6, which records the failed state in variables $fi = i, fj = j$ and $fid = id$. At line 21 of Algorithm 5.4, after checking some conditions, an `EDITOR()` presented in Algorithm 5.10 is called by the Urdu parser to heal this discontinuous state.

When the parser is discontinued at $chart[i+1]$, then by using recorded variables in `SCANNER()`, the `EDITOR()` first deletes and re-initializes the $chart[i+1]$. In a sequence, it further deletes and re-initializes the charts from i to $fi+1$. fi is basically the chart number, where homonyms are available and fj is the location of that wrong choice e.g. CM → کی in a chart. The wrong choice is stored in a *rule* array after splitting by space. A loop is used to traverse all productions of a chart where the homonyms are

Algorithm 5.10 Editor

```
1: function EDITOR(i, id, fi, fj, fid, chart, chartSize)
2:     Drop and re-initialize chart[i + 1]
3:     for z ← i to fi+1 step -1 do
4:         Drop and re-initialize chart[z]
5:     end for
6:     rule ← chart[fi].getRule(fj).split(" ")          ▷ splitting rule with space
7:     for z ← 0 to chartSize[fi]-1 do
8:         temprule ← chart[fi].getRule(z).split(" ")
9:         if temprule[2] = rule[2] then
10:            if !(temprule[0] = rule[0]) then
11:                j ← z − 1, i ← fi, id ← z
12:                break
13:            end if
14:        end if
15:    end for
16: end function
```

available. The chart productions are stored temporarily in a *temprule* array. When the RHS of both the arrays are found to be equal then the LHS are checked. If LHS are found to be equal then those productions are skipped and recorded otherwise. In this way, the next choice V.PERF → کی is located and the i_{th} and j_{th} loop variables of the Urdu parser are set to that choice for further processing. Continuing in this way, the parser finally moves towards an optimal solution.

5.6.7 Limitations of Completer

This issue of wrong solution or a discontinuous/failed parse obtained during parsing is related to subordinate clauses, when the number of subordinate clauses in a sentence is more than one. The issue does not appear often and is related to NL type productions, specially the subordinate clauses denoted by SBAR in an annotated or a parsed sentence. The wrong production caused the parsing discontinuity can lie in any chart but one thing is sure that the right choice of production also exists in the same chart but not selected by the `Completer` due to the order of productions. In the Earley algorithm, after finalizing its choice of production, the `Completer` never comes back to seek

M.S →	- @
SBAR →	C.SBORD NP.NOM-SUB SBAR ADVP-MNR-MODF NP.NOM-MNR-OBJ VCMAIN M.S @
SBAR →	C.SBORD NP.NOM-SUB SBAR @ ADVP-MNR-MODF NP.NOM-MNR-OBJ VCMAIN M.S
ADVP-MNR-MODF →	@ ADV.MNR

Figure 5.7: Incomplete solution of the Urdu paser

the other choice even made a wrong choice in its first attempt. Due to this restriction, it was observed that the parser produced a discontinuous parse in case of the nested subordinate clauses and the discontinuity occurred during the processing of the last chart when the production of subordinate clause SBAR was being manipulated.

(5.6) *[In laRAyI jHagRON=kI vajah=sE Ap=kO*
[these.Pl fight.Fem dispute.Obl=Poss reason.Fem.Sg=Inst he=Dat
moqa2h nah mil sakA [kEh Ap
opportunity.Masc.Sg not find.Root can.Perf [**that/SBAR** he=Nom
[jEsA dil cAhtA tHA us t2arah
[**as/SBAR** heart want.Imperf.Masc.Sg be.Past.Masc.Sg that like
is3lAh2 farmAEN .]]]
reforms do.Subj full-stop]]]
'Due to these conflicts, he could not find an opportunity that he could do the reforms like that as his heart wanted '

A sentence of 23 tokens in 5.6 with two subordinate clauses highlighted with SBAR is also an evidence of this issue among the other examples. After the parsing of productions in the last (23rd) chart for this sentence, the order of the complete productions should be as follows. The '@' at the end of each production represents the complete status of productions.

M.S →	- @
SBAR →	C.SBORD NP.NOM-SUB **SBAR** ADVP-MNR-MODF NP.NOM-MNR-OBJ VCMAIN M.S @
S →	KP-INST-MODF KP.DAT-SUB NP.NOM-OBJ VCMAIN **SBAR** @

But, unfortunately, the parser went into a discontinuous state without any solution during processing of the last chart with the productions given in Figure 5.7. Which means that the parser did not find a complete solution production 'S' with '@' at its end. As it was the last chart processed for the given sentence and another chart could

not be added. So, the parser discontinued finally with the incomplete productions in the last chart without any solution.

In Figure 5.7, the parser performed well up to the completion of first SBAR → ... @ production. Then the `COMPLETER()` went back in charts to search a related incomplete production with the properties like SBAR on its RHS and the '@' symbol before the SBAR, as @ SBAR. The `COMPLETER()` made a fault in chart 12 in the presence of a wrong choice with the similar properties at higher precedence. It found an incomplete SBAR production as SBAR → C.SBORD NP.NOM-SUB **@ SBAR** ADVP-MNR-MODF NP.NOM-MNR-OBJ VCMAIN M.S. As the '@' symbol was before the SBAR, so the `COMPLETER()` picked it up. After moving the '@' forward, it added the updated production in the last chart as can be seen in Figure 5.7. The `PREDICTOR()` then checked the '@' before the ADVP-MNR-MODF and the parser went into a wrong direction. It happened because the `COMPLETER()` picked up the wrong production in chart 12, even the right choice S → KP-INST-MODF KP.DAT-SUB NP.NOM-OBJ VCMAIN @ SBAR was also there in the early rows of that chart. To resolve this issue, it is needed to allow the `COMPLETER()` to go back further until a successful parse has been obtained as can be seen in Algorithm 5.11.

When the Urdu parser called the `COMPLETER()`, it first sets the `completerCheck` flag to false. This is the flag which is used to allow the `COMPLETER()` to back track the right choice among the NL type productions. The `COMPLETER()` has two main loops. The external loop is used to handle the previous charts $chart[pc]$ from $i-1$ to 0. The internal loop is used to traverse the chart states from 0 to $chartSize[pc]-1$. With the execution of internal loop, each candidate chart production $chart[pc]$.getRule(c) is stored first in the *previousRule* array through the *dummyString*. An already processed position of the '@' symbol is recorded in the *dummy@Position*. After getting index of the '@' from the *previousRule*, a condition is checked that the production in the *previousRule* is not a complete production. If it is the case then after matching the LHS of the current chart production *currentRule*[0] with the non-terminal after the '@' in the previous chart production *previousRule*.get($pIndex$+1), the '@' symbol is moved forward. The updated and formatted *previousRule* is stored in another array *pStr* for further processing. The existence of this updated production in $chart[i]$ is checked via a *recursion* boolean variable. If the *recursion* is found to be false then back-pointers are calculated first as explained in Section 5.6.3. After calculating the back-pointers, the updated production entry is then added and printed into a $chart[i]$ by setting the `completerCheck` to true. After updating some variable, the *previousRule*

Algorithm 5.11 Completer

```
1: function COMPLETER(i, j, chartSize, chart, currentRule, id, Sentence)
2:     completerCheck ← false                          ▷ completerCheck flag set to false
3:     for pc ← i-1 To 0 do                            ▷ loop to handle previous charts
4:         for c ← 0 To chartSize[pc]-1 do             ▷ for traversing chart entries
5:             dummyString ← chart[pc].getRule(c).split(" ")        ▷ splitting with space
6:             dummy@Position ← chart[pc].get@Position(c).split(",")  ▷ splitting with ','
7:             previousRule.add(dummyString[])     ▷ storing dummyString in a list
8:             pIndex ← previousRule.indexOf("@")          ▷ locating '@' position
9:             if pIndex+1 ≠ SIZE(previousRule) then
10:                if currentRule[0] = previousRule.get(pIndex+1) then
11:                    move '@' forward and format previousRule, then store it in pStr
12:                    recursion ← false
13:                    Check pStr in chart[i], if exist then recursion ← true
14:                    if !recursion then
15:                        call BACKPOINTER()
16:                        chart[i].add("id", pStr, k+","+i, backPointer, "Completer" )
17:                        completerCheck ← true
18:                        chart[i].print(StateId, Rule, @Position, BackPointer, Operation)
19:                        id = id + 1                          ▷ increment StateId
20:                        chartSize[i] ← id                    ▷ chartSize is updated
21:                        previousRule.clear()
22:                    end if
23:                else
24:                    previousRule.clear()
25:                end if
26:            else
27:                previousRule.clear()
28:            end if
29:        end for
30:        if completerCheck = true & i < LENGTH(Sentence[]) then
31:            break      ▷ pc loop exit because completer() has finished and no back track
32:        else if completerCheck = true & i ≥ LENGTH(Sentence[]) then
33:            completerCheck ← false
34:        end if
35:    end for
36: end function
```

is again cleared and the internal loop continues. However, if the *previousRule* is found to be a complete state, then the array *previousRule* is made empty and the internal loop continues. At the end of internal loop, there are some conditions which deal with the discontinuity issues of the `COMPLETER()` discussed in this section. If the *completerCheck* is found to be true and the chart number is less than the length of the sentence then the external loop breaks, which means that the `COMPLETER()` has found the solution and there is no need to go back. On the other hand, if the *completerCheck* is found to be true and the chart number is greater or equal to the length of the sentence then the `COMPLETER()` is allowed to back track by setting its flag to its default value.

5.6.8 A Run Example

To see the effect of the Urdu parser, we examine example 5.7. The length of the sentence is five tokens, for which the parser generates 0 to 5 charts. All irrelevant NL type productions are removed from the output charts to save the document space. Due to this removal, the values in the STATID column of Figure 5.8 are not consecutive. The charts generated through the parser without the bracketed parse tree are displayed in Figure 5.8. After displaying these charts, the parser then displays the bracketed parse tree of this sentence, which can be seen in Figure 5.9.

(5.7) قبلائی خان منگول تھا ۔

QublAI	*xAn*	*mangOl*	*tHA*	-
N.PROP	N.PROP	N.PROP	V.COP.PAST	

'Kublai Khan was a Mangol.'

After the removal of issues discussed in Section 5.6, the parser has got the following benefits. The output of the parser is so directed, speedy and refined in the sense that no extra or irrelevant L type productions are introduced by the `PREDICTOR()`. It is really hard now that the `SCANNER()` will select a wrong choice of production. If it happens then the `SCANNER()` has a tendency to correct itself. The `COMPLETER()` is boosted up to find a solution if the solution is available in any chart at any position. Empty productions are handled but it is better to remove them from the grammar completely. The calculation of back-pointers in the `COMPLETER()` is helpful to the `BUILDER()`, which builds bracketed parse trees of the given sentence. The `EDITOR()` provides a mechanism through which failed parsed output is edited and corrected. These all features make up an Urdu parser with rich morphological, syntactical and functional information in the form of a grammar in it.

CHART(0)				
STATEID	**RULE**	**DOT-POSITON**	**BACK-POINTER**	**OPERATION**
0	ROOT @ S	0,0		Seed
17	S @ NP.NOM-SUB NP.NOM-PLINK VCMAIN M.S	0,0		Predictor
57	NP.NOM-SUB @ N.PROP N.PROP	0,0		Predictor
119	N.PROP @ قبلائی	0,0		Predictor
CHART(1)=قبلائی				
STATEID	**RULE**	**DOT-POSITON**	**BACK-POINTER**	**OPERATION**
0	N.PROP قبلائی @	0,1		Scanner
1	NP.NOM-SUB N.PROP @ N.PROP	0,1	1-0	Completer
4	N.PROP @ خان	1,1		Predictor
CHART(2)=خان				
STATEID	**RULE**	**DOT-POSITON**	**BACK-POINTER**	**OPERATION**
0	N.PROP خان @	1,2		Scanner
1	NP.NOM-SUB N.PROP N.PROP @	0,2	1-0 2-0	Completer
5	S NP.NOM-SUB @ NP.NOM-PLINK VCMAIN M.S	0,2	2-1	Completer
35	NP.NOM-PLINK @ N.PROP	2,2		Predictor
90	N.PROP @ منگول	2,2		Predictor
CHART(3)=منگول				
STATEID	**RULE**	**DOT-POSITON**	**BACK-POINTER**	**OPERATION**
0	N.PROP منگول @	2,3		Scanner
1	NP.NOM-PLINK N.PROP @	2,3	3-0	Completer
5	S NP.NOM-SUB NP.NOM-PLINK @ VCMAIN M.S	0,3	2-1 3-1	Completer
17	VCMAIN @ V.COP.PAST	3,3		Predictor
46	V.COP.PAST @ تھا	3,3		Predictor
CHART(4)=تھا				
STATEID	**RULE**	**DOT-POSITON**	**BACK-POINTER**	**OPERATION**
0	V.COP.PAST تھا @	3,4		Scanner
1	VCMAIN V.COP.PAST @	3,4	4-0	Completer
2	S NP.NOM-SUB NP.NOM-PLINK VCMAIN @ M.S	0,4	2-1 3-1 4-1	Completer
4	M.S @ -	4,4		Predictor
CHART(5)=-				
STATEID	**RULE**	**DOT-POSITON**	**BACK-POINTER**	**OPERATION**
0	M.S - @	4,5		Scanner
1	S NP.NOM-SUB NP.NOM-PLINK VCMAIN M.S @	0,5	2-1 3-1 4-1 5-0	Completer
2	ROOT S @	0,5	5-1	Completer

Figure 5.8: Charts generated by the Urdu parser for a sentence 5.7

5.7 Results

The URDU.KON-TB treebank contains 1400 annotated sentences. The sentences are divided into 80% training data (for the development of grammar) and 20% test data (for the development of test suite). First, a CFG is extracted computationally from the training data which is categorized into L and NL type productions. The test data is further divided into 10% held-out data and 10% test data. The held-out data is used in the development of the Urdu parser, while the test data is used for evaluation of results after the completion of the Urdu parser. To make the test data more valuable and reliable for results, ten sentences from each hundred of 1400 sentences are selected. The test data so contains 140 sentences in all. In the test data, the average length is 13.73 words per sentence, the minimum length of a sentence found is 5 words and

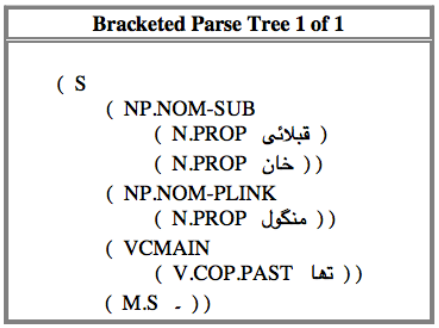


Figure 5.9: A bracketed parse tree generated by the Urdu parser for a sentence 5.7

the maximum length of a sentence is 46 words. All items which can exist in a normal text are considered e.g., punctuation, null elements, diacritics, headings, regard titles, Hadees (statements of the prophets), anaphora within a sentence, and others except the unknown[1] words which will be dealt statistically when this parser will be shifted towards a probabilistic one.

The PARSEVAL measures are used to evaluate the results of the Urdu parser. The evaluation results of the Urdu parser depicted in Table 5.1 includes the number of sentences in test, gold and matched set, precision and recall percentages, f-score, and crossing brackets. The PARSEVAL measures are calculated in two ways on the basis of constituents and their results are presented in rows as A-1 and T-2. The results are calculated through a module developed in the Perl programming environment. The results in row A-1 of the table are calculated on the average basis and the results in row T-2 are calculated on the basis of totality, which is a standard method of calculation.

	Sentences	Length	Matched	Gold	Test	Precision	Recall	F-score	Crossing
A-1	140	13.73	17	22	18	0.952	0.811	0.848	2
T-2	140	1922	2449	3107	2531	0.968	0.788	0.869	329

Table 5.1: Evaluation results of the Urdu parser

A two stage constraint based dependency parser (CBP) for Hindi (Bharati et al., 2008) was trained on 1185 sentences from the Hindi treebank (Begum et al., 2008). The test data contained 220 sentences. The evaluation results in the form of unlabeled attachment (UA), labeled (L) and labeled attachment (LA) are provided by Bharati

[1]Any word which is not listed in the lexicon of the Urdu parser

et al. (2008). From which only the percentage values for LA can be compared with the results of the Urdu parser because LA counts only the correct number of heads and dependency arcs with its labels. The percentage value of LA reported without recall is 75%. However, the LA with recall (LA-R) percentages are also provided for 12 different labels, whose average is 66.32%. As the LA counts only the correct number of heads and dependency arcs with its labels or constituents produced in the parse, so according to this the Urdu parser has the constituent accuracy or LA in T-2 as 96.76%[1], which is 21.76% better than the two stage Hindi dependency parser. On the other hand, for comparison of the LA-R, we can compare the calculated average percentage 66.32% with the recall value in A-1 of the Table 5.1 logically, but being on the safe side after comparing this value with the recall percentage in T-2, it can be concluded that the Urdu parser outperforms the two stage Hindi dependency parser with 12.48% in recall.

The Urdu parser outperforms the simple Hindi dependency parser by Bharati et al. (2009a) with an additional recall of 22%. In (Bharati et al., 2009a), only precision and recall percentages are given. That's why only the precision and recall percentages of labeled attachment (LA) are compared. For chunks, intra-chunks and karakas, the precision percentages of LA (LA-P) achieved by the simple Hindi dependency parser are 82.3%, 71.2% and 74.1%, respectively. The average of these LA-P percentages is 75.9%, which is 20.9% less precision than the Urdu parser in row T-2. Similarly, Hindi dependency parser achieved LA recalls in case of chunks, intra-chunks and karakas as 65.4%, 58.2% and 46.7% respectively. The average of these percentages is calculated as 56.8%, which is now the final LA recall percentage of the Hindi dependency parser. For comparison, the recall percentage of the Urdu parser used is mentioned in row T-2 as 78.8%. The values obtained for both the parsers concludes that the Urdu parser outperforms the simple Hindi dependency parser with 22% increase in recall.

Multi-path shift-reduce parser (Mukhtar et al., 2012b) for Urdu parsed 74 sentences successfully out of 100 and it was then reported as a 74% of accuracy. This evaluation is very weak because the successful parsed sentences were not compared with the gold standard. *Recall* is a value obtained through dividing the *Matched* constituents with the constituents available in *Gold* data. As recall percentage in our case is 78.8%, so we can say that the Urdu parser beats the multi-path shift-reduce parser with a 4.8% increase in recall. On the other hand, from the first 100 sentences of the test data, the Urdu parser provides 89 sentences with parsed solutions. Comparatively, the Urdu parser has 15% more accuracy than the Multi-path shift-reduce parser, but the parsed

[1]Getting percentage by dividing the correct/matched constituents with the constituents produced in the test parse by the Urdu parser as ((2449x100)/2531)

solutions are not compared with the *Gold* data. So, by considering the safe side, we can repeat our argument that the Urdu parser beats the multi-path shift-reduce parser with a 4.8% increase in recall.

The results given in Table 5.1 generally describe that for the *Length* of a sentence in words, the parser produces a number of constituents in a *Test* parse. If *Test* parse contains such number of constituents then such number of constituents are *Matched* with the constituents available in the *Gold* data. *Precision* is calculated by dividing the *Matched* constituents with the constituents available in the *Test* parse, while the *Recall* is calculated by dividing the *Matched* constituents with the constituents available in the *Gold* data. The *F-score* is the harmonic mean of *Precision* and *Recall*.

5.8 Summary

By considering the state-of-the-art requirements mentioned in Section 5.1, a grammar having rich morphological, syntactical and functional information is extracted from the URDU.KON-TB treebank. The existing dynamic programming algorithm known as the Earley algorithm was extended to fulfill our parsing requirements. Several Urdu parsing issues are resolved through this extension. The parser with rich morphological information is comparable or better than the state-of-the-art. The evaluation results are compared with three parsers which includes the multi-path shift-reduce parser for Urdu, a two stage constraint based dependency parser for Hindi and the simple Hindi dependency parsers. The Urdu parser has better evaluation results as compared to the other three parsers presented. The parser can help the linguists to analyze the Urdu sentences computationally and it can be useful in Urdu language processing and machine learning domains. By using this parser, the size of the treebank can also be increased. At present, the treebank contains only 1400 annotated sentences, which is needed to be increased. This can be done after getting partial parsed trees of unknown sentences. These partial parsed trees can be corrected and then imported into the URDU.KON-TB treebank. Using this methodology, one can speedily increase the size of treebank. The issues discussed in Section 5.6.5 about handling of empty productions will be addressed further, when this parser will be shifted towards statistical one. At present, the `COMPLETER()` is dealing only those discontinuity problems which occur during the process of the last chart. So, the `COMPLETER()` of the parser will be enhanced to deal with the discontinuity problem which can occur before the last chart.

6

Conclusion

In this thesis, the development of the URDU.KON-TB treebank and parser for the under-resourced language Urdu has been described. The URDU.KON-TB treebank using phrase structure and hyper dependency structure annotation along with the semi-semantic part of speech, semi-semantic syntactic and functional tag sets was presented in chapter 3. The annotation of the URDU.KON-TB treebank whose annotation guidelines are given in Chapter 2 was evaluated statistically. The statistical model used for this purpose is known as Krippendorff's α co-efficient. The task of evaluation performed along with the error analysis was presented in Chapter 4. Finally, a grammar built from this treebank was then given to the Urdu parser. The development of this parser along with its issues is discussed in Chapter 5. The summary and conclusion of each respective chapter in this thesis is presented as follows.

After establishing a general introduction of the whole work in Chapter 1, the presentation of the annotation guidelines in Chapter 2 is given. Chapter 2 detailed an up to date annotation guideline for the URDU.KON-TB treebank. It included instructions for the POS, syntactic and functional annotations. Section 2.1 detailed the annotation instructions regarding semi-semantic POS tagging. The syntactic annotation instructions are outlined in detail in Section 2.2. The semantic features which can travel from the lexical level to the syntactic level were discussed in Section 2.3 along with the functional annotation of the URDU.KON-TB treebank. All the sections contained annotated examples for each category and its subcategories. The semantic features include temporal, spatial, instrumental, etc. and the functional features include subject, object, oblique, etc. along with some labels for clauses and anaphora resolution.

Chapter 3 detailed the development process of the URDU.KON-TB treebank for the South Asian language Urdu. The construction of the URDU.KON-TB treebank was

based on a corpus, which was collected from Urdu Wikipedia and Jang newspaper. The corpus contains a blend of text from local & international news, social stories, sports, culture, finance, religion, traveling, etc. A combination of a phrase structure and a hyper dependency structure is practiced in the manual annotation of the URDU.KON-TB treebank. The treebank is encoded with sufficient morphological, part of speech, syntactical and functional information, which is necessary for the parsing of a morphologically rich language like Urdu. The resources developed are a semi-semantic part of speech tag set, a semi-semantic syntactic tag set, a functional tag set and a bracketed annotated corpus named as the URDU.KON-TB treebank. The main POS tag set contained twenty two (22) tags, which are divided into further morphological and semantical subcategories. Similarly, the main syntactic tag set contained twenty six (26) tags, which are divided into further syntactical and semantical subcategories. Finally, the functional tag set included eighteen (18) tags. The URDU.KON-TB treebank contains 1400 annotated sentences. The size of the URDU.KON-TB treebank as developed within the scope of this thesis is limited. However, the size of the URDU.KON-TB treebank can be increased by using the Urdu parser that was developed as part of this thesis.

Chapter 4 presented the annotation evaluation and a subsequent revision of the annotation scheme. As reliable annotation of a corpus is a necessity in corpus linguistics, so, after the investigation of related work and our requirements, the Krippendorff's α coefficient was selected for the evaluation of our annotation scheme. A proper selection of native annotators and their training was conducted at the University of Sargodha, Pakistan. The annotation of the semi-semantic POS, semi-semantic syntactic and functional tags was evaluated according to the norms of Krippendorff's α coefficient. For this annotation, one hundred raw and random sentences collected from the URDU.KON-TB treebank were given to five annotators with the annotation guidelines of the treebank. The α values of 0.964, 0.817 and 0.806 for an inter-annotator agreement were found for the semi-semantic POS, semi-semantic syntactic and functional annotations, respectively. These values lie in the range of perfect agreement according to a scale. Error analysis was performed and the issues that emerged during the training course of annotation evaluation led to a revision of the tag sets. The issues resolved and the issues still pending (with or without proposals) are discussed in detail. In the process of resolving the issues, the tags renewed or removed from the annotation guidelines are also presented. With this exercise of annotation evaluation, a standard annotation guidelines for the URDU.KON-TB treebank were finalized, which are presented in Chapter 2.

The development of an automatic Urdu parser is presented in Chapter 5. In this chapter, a computationally extracted context free grammar from the URDU.KON-TB treebank was divided into 80% training data and 20% test data. The test data was further divided into 10% held out data and 10% test data. The held out data was used during the development of the Urdu parser, while the test data was used to evaluate the results of the Urdu parser after its development. The first ten sentences from each hundred of 1400 sentences were selected. The test data thus contains 140 sentences in all. In the test data, the minimum, average and maximum length of a sentence is 5, 13.73 and 46 words per sentence respectively. All items which can exist in a normal text are considered e.g. punctuation, null elements, diacritics, headings, regard titles, Hadees (the statements of prophets), anaphora within a sentence, etc. The Urdu parser is an extended version of the Earley parsing algorithm. This Urdu parser with rich morphological, POS, syntactical and functional information is comparable or better than the state-of-the-art in Urdu language processing. The parser performs better than the multi-path-shift-reduce parser, a two stage Hindi dependency parser and a simple Hindi dependency parser. After the removal of the issues faced during the development of the Urdu parser, the output of the parser is so much directed, speedy and refined in a sense that no extra or irrelevant L type productions are introduced by the `PREDICTOR()`. It is really hard now that the `SCANNER()` will select a wrong choice of production. If it happens then the `SCANNER()` has a tendency to correct itself. The `COMPLETER()` is boosted up to find a solution, if the solution is available in any chart at any position. The calculation of back pointers is helpful for the `BUILDER()`, which builds bracketed parse trees. The `EDITOR()` provides a mechanism through which failed parsed output is automatically edited and corrected. These all features concludes an Urdu parser with rich morphological, POS, syntactical and functional information in the form of a grammar in it.

Overall, this thesis reported the development of the two major computational resources for the South Asian language Urdu, which includes the URDU.KON-TB treebank and the Urdu parser. The quality of the URDU.KON-TB treebank is strengthened with the development of its annotation guidelines and annotation reliability evaluation via Krippendorff's α coefficient. The URDU.KON-TB treebank can be used for probabilistic parsing, training of part of speech taggers, disambiguation of spoken sentences, grammar development, sources for linguistic inquiry, psychological modeling, pattern matching and in many other applications. The URDU.KON-TB treebank is encoded with sufficient morphological, part of speech, syntactical and functional information, which is necessary for the parsing of a morphologically rich language like Urdu. The

Urdu parser developed can help linguists analyze Urdu sentences computationally and it can be useful in Urdu language processing and machine learning domains. The size of the URDU.KON-TB treebank can be increased by getting the partial parsed trees of unknown sentences through this parser. These partial parsed trees can be corrected and then integrated into the URDU.KON-TB treebank. This Urdu parser is language independent and can be used for other languages. Only thing which is required is the context free grammar of that language to be parsed.

As a future work, the issues still lying in the annotation of the URDU.KON-TB treebank can be resolved and the annotation guidelines can be revised after the annotation evaluation via the Krippendorff's α coefficient. After updating the URDU.KON-TB treebank with the revised annotation, the grammar can be extracted again for the Urdu parser to parse the test data. The difference with respect to the recent reported values of the annotation evaluation and the parsing evaluation in this thesis can be reported in future. The context free grammar of the Urdu parser can be converted into probabilistic context free grammar and in this way, the rule based model of the Urdu parser can be shifted towards the probability based model. The issues discussed in Chapter 5 about the handling of empty productions can be addressed further. The `COMPLETER()` of the parser can be enhanced to deal the discontinuity problems which can occur before the last chart. As the Urdu parser is language independent, so in parallel to the URDU.KON-TB treebank, further treebanks for other local languages of Pakistan like Punjabi, Saraiki, Hindko, etc, can be constructed easily because some languages like mentioned are very much close in structure to Urdu. The grammars then can be extracted from these treebanks to parse the languages on the Urdu parser. Moreover, the parallel treebanks that can be developed through this exercise will become a good source for machine translation among these languages.

References

The American Heritage New Dictionary Of Cultural Literacy, Third Edition, Mar 2014. URL http://dictionary.reference.com/browse/subjunctive. 23

B. Aarts, S. Chalker, and E. Weiner. *The Oxford Dictionary Of English Grammar*. Oxford University Press, 2014. 6, 8, 9, 11, 19, 24

Q. Abbas. Building A Hierarchical Annotated Corpus Of Urdu: The URDU.KON-TB Treebank. *Lecture Notes in Computer Science*, 7181(1):66–79, 2012. doi: 10.1007/978-3-642-28604-9_6. URL http://dx.doi.org/10.1007/978-3-642-28604-9_6. 1, 60, 64, 72, 73, 103, 125, 132, 141, 156, 159, 162

Q. Abbas. Semi-Semantic Part Of Speech Annotation And Evaluation. In *Proceedings of ACL 8th Linguistic Annotation Workshop held in conjunction with COLING*, pages 75–81, Dublin, Ireland, August 2014a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W14-4911. 74

Q. Abbas. Exploiting Language Variants Via Grammar Parsing Having Morphologically Rich Information. In *Proceedings of the EMNLP Workshop on Language Technology for Closely Related Languages and Language Variants*, pages 35–45, Doha, Qatar, October 2014b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W14/W14-4204. 155

Q. Abbas and A. Nabi Khan. Lexical Functional Grammar For Urdu Modal Verbs. In *Emerging Technologies, 2009. ICET 2009. International Conference on*, pages 7–12. IEEE, 2009. 22, 25, 86, 141

Q. Abbas and G. Raza. A Computational Classification Of Urdu Dynamic Copula Verb. *International Journal of Computer Applications*, 85(10):1–12, January 2014. 19

REFERENCES

Q. Abbas, N. Karamat, and S. Niazi. Development Of Tree-Bank Based Probabilistic Grammar For Urdu Language. *International Journal of Electrical & Computer Science*, 9(09):231–235, 2009. 3, 68, 98, 156, 159, 160

Q. Abbas, M. Ahmed, and S. Niazi. Language Identifier For Languages Of Pakistan Including Arabic And Persian. *International Journal of Computational Linguistics (IJCL)*, 1(03):27–35, 2010. 100

A. Abbi. *Reduplication In South Asian Languages: An Areal, Typological, And Historical Study*. Allied Publishers New Delhi, 1992. 6, 78

B. Agrawal, R. Agarwal, S. Husain, and D. M. Sharma. An Automatic Approach To Treebank Error Detection Using A Dependency Parser. In *Computational Linguistics and Intelligent Text Processing*, pages 294–303. Springer, 2013. 160

T. Ahmed and M. Butt. Discovering Semantic Classes For Urdu NV Complex Predicates. In *Proceedings of the Ninth International Conference on Computational Semantics*, pages 305–309. Association for Computational Linguistics, 2011. 20, 88

A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman. *Compilers: Principles, Techniques, & Tools*, volume 1009. Pearson/Addison Wesley, 2007. 182

W. Ali and S. Hussain. Urdu Dependency Parser: A Data-Driven Approach. In *Proceedings of Conference on Language and Technology (CLT10)*, 2010. 159, 160

A. W. Appel and J. Palsberg. *Modern Compiler Implementation In Java*. Cambridge University Press New York, 2007. 182

R. Artstein and M. Poesio. Inter-Coder Agreement For Computational Linguistics. *Computational Linguistics*, 34(4):555–596, 2008. 101, 102, 103, 108

A. Arun and F. Keller. Lexicalization In Crosslinguistic Probabilistic Parsing: The Case Of French. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 306–313. Association for Computational Linguistics, 2005. 1, 155

J. Aycock and R. N. Horspool. Practical Earley Parsing. *The Computer Journal*, 45 (6):620–630, 2002. 182, 183, 184

R. Begum, S. Husain, A. Dhwaj, D. M. Sharma, L. Bai, and R. Sangal. Dependency Annotation Scheme For Indian Languages. In *IJCNLP*, pages 721–726, 2008. 159, 160, 192

B. Bennett. 334. Note: Measures For Clinicians' Disagreements Over Signs. *Biometrics*, pages 607–612, 1972. 102

E. M. Bennett, R. Alpert, and A. Goldstein. Communications Through Limited-Response Questioning. *Public Opinion Quarterly*, 18(3):303–308, 1954. 101

A. Bharati, V. Chaitanya, R. Sangal, and K. Ramakrishnamacharyulu. *Natural Language Processing: A Paninian Perspective*. Prentice-Hall of India New Delhi, 1995. 69, 160

A. Bharati, M. Bhatia, V. Chaitanya, and R. Sangal. Paninian Grammar Framework Applied To English. Technical report, Technical Report TRCS-96-238, CSE, IIT Kanpur, 1996. 159

A. Bharati, S. Husain, D. M. Sharma, and R. Sangal. A Two-Stage Constraint Based Dependency Parser For Free Word Order Languages. In *Proceedings of the COLIPS International Conference on Asian Language Processing 2008 (IALP)*, 2008. 159, 160, 192

A. Bharati, M. Gupta, V. Yadav, K. Gali, and D. M. Sharma. Simple Parser For Indian Languages In A Dependency Framework. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 162–165. Association for Computational Linguistics, 2009a. 159, 160, 162, 193

A. Bharati, D. M. Sharma, S. Husain, L. Bai, R. Begam, and R. Sangal. Anncorra: Treebanks For Indian Languages, Guidelines For Annotating Hindi Treebank, 2009b. 69

R. A. Bhat and D. M. Sharma. A Dependency Treebank Of Urdu And Its Evaluation. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 157–165. Association for Computational Linguistics, 2012a. 104, 110

R. A. Bhat, S. Jain, and D. M. Sharma. Experiments On Dependency Parsing Of Urdu. In *In Proceedings of The 11th International Workshop on Treebanks and Linguistic Theories (TLT11)*, 2012b. 159, 160

R. Bhatt, B. Narasimhan, M. Palmer, O. Rambow, D. M. Sharma, and F. Xia. A Multi-Representational And Multi-Layered Treebank For Hindi/urdu. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 186–189. Association for Computational Linguistics, 2009. 69

REFERENCES

R. Bhatt, T. Bögel, M. Butt, A. Hautli, S. Sulger, and T. H. King. Urdu/Hindi Modals. In *Proceedings of the LFG11 Conference. Hong Kong*, 2011. 25, 86, 141

A. Bies, M. Ferguson, K. Katz, R. MacIntyre, V. Tredinnick, G. Kim, M. A. Marcinkiewicz, and B. Schasberger. Bracketing Guidelines For Treebank II Style Penn Treebank Project. *University of Pennsylvania*, 97, 1995. 100

S. Black, Abney, S. Flickenger, C. Gdaniec, C. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. 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. 104

T. Bögel and M. Butt. Possessive Clitics And Ezafe In Urdu. *Morphosyntactic Categories and the Expression of Possession*, 199:291, 2013. 86, 129

T. Bögel, M. Butt, A. Hautli, and S. Sulger. Developing A Finite-State Morphological Analyzer For Urdu And Hindi. *Finite State Methods and Natural Language Processing*, page 86, 2007. 6, 78

T. Bögel, M. Butt, A. Hautli, and S. Sulger. Urdu And The Modular Architecture Of ParGram. In *Proceedings of the Conference on Language and Technology*, 2009. 70

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. The TIGER Treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, pages 24–41, 2002. 66, 105

T. Brants. Inter-Annotator Agreement For A German Newspaper Corpus. In *LREC*, 2000a. 104

T. Brants. TnT: A Statistical Part-Of-Speech Tagger. In *Proceedings of the sixth conference on Applied natural language processing*, pages 224–231. Association for Computational Linguistics, 2000b. 104

T. Brants and O. Plaehn. Interactive Corpus Annotation. In *LREC*, 2000. 104

R. L. Brennan and D. J. Prediger. Coefficient Kappa: Some Uses, Misuses, And Alternatives. *Educational and psychological measurement*, 41(3):687–699, 1981. 102

E. Brill. Discovering The Lexical Features Of A Language. In *Proceedings of the 29th annual meeting on Association for Computational Linguistics*, pages 339–340. Association for Computational Linguistics, 1991. 66

E. Brill, D. Magerman, M. Marcus, and B. Santorini. Deducing Linguistic Structure From The Statistics Of Large Corpora. In *Information Technology, 1990.'Next Decade in Information Technology', Proceedings of the 5th Jerusalem Conference on (Cat. No. 90TH0326-9)*, pages 380–389. IEEE, 1990. 66

L. J. Brinton and D. Brinton. *The Linguistic Structure Of Modern English*. John Benjamins Publishing, 2010. 62

R. Bruce, J. Wiebe, et al. Word Sense Distinguishability And Inter-Coder Agreement. In *Proc. 3rd conference on Empirical methods in Natural Language Processing (EMNLP-98)*, pages 53–60, 1998. 102

M. Butt. *The Structure Of Complex Predicates In Urdu*. Center for the Study of Language (CSLI), 1995. 19, 88, 93, 138

M. Butt. The Light Verb Jungle. In *Workshop on Multi-Verb Constructions*, 2003. 20, 21

M. Butt. *Theories Of Case*. Cambridge University Press, 2006. 70, 84

M. Butt. The Light Verb Jungle: Still Hacking Away. *Complex predicates: cross-linguistic perspectives on event structure*, page 48, 2010. 79, 80, 86, 88, 126, 131

M. Butt and T. Ahmed. The Redevelopment Of Indo-Aryan Case Systems From A Lexical Semantic Perspective. *Morphology*, 21(3-4):545–572, 2011. 85

M. Butt and T. H. King. Case Systems: Beyond Structural Distinctions. *New perspectives on case theory*, pages 53–87, 2003. 86

M. Butt and T. H. King. The Status Of Case. In *Clause structure in South Asian languages*, pages 153–198. Springer, 2004. 11, 29, 34, 38, 68, 75, 85, 86, 90, 146, 151, 153

M. Butt and T. H. King. Urdu In A Parallel Grammar Development Environment. *Language Resources and Evaluation*, 41(2):191–207, 2007. 70, 85, 159

M. Butt and G. Ramchand. Complex Aspectual Structure In Hindi/Urdu. *M. Liakata, B. Jensen, & D. Maillat, Eds*, pages 1–30, 2001. 79, 81

M. Butt and J. Rizvi. Tense And Aspect In Urdu. *Layers of Aspect. Stanford: CSLI Publications*, 2010. 79

M. Butt and L. Sadler. Verbal Morphology And Agreement In Urdu. *Syntactic structures and morphological information. Mouton*, pages 57–100, 2003. 11

M. Butt, T. King, F. Segond, and M.-E. Nino. *A Grammar Writer's Cookbook*, volume 95. CSLI publications Stanford, 1999. 60, 62, 71, 95

M. Butt, T. H. King, and S. Roth. Urdu Correlatives: Theoretical And Implementational Issues. In *Proceedings of the LFG07 Conference*, pages 107–127. Citeseer, 2007. 56

J. Carletta. Assessing Agreement On Classification Tasks: The Kappa Statistic. *Computational linguistics*, 22(2):249–254, 1996. 104, 114

J. Carletta, S. Isard, G. Doherty-Sneddon, A. Isard, J. C. Kowtko, and A. H. Anderson. The Reliability Of A Dialogue Structure Coding Scheme. *Computational linguistics*, 23(1):13–31, 1997. 102, 103, 114

L. Carlson, D. Marcu, and M. E. Okurowski. *Building A Discourse-Tagged Corpus In The Framework Of Rhetorical Structure Theory*. Springer, 2003. 102, 114

N. Chomsky. Three Models For The Description Of Language. *Information Theory, IRE Transactions on*, 2(3):113–124, 1956. 65

N. Chomsky. *Lectures On Government And Binding: The Pisa Lectures*, volume 9. Walter de Gruyter, 1993. 69

D. V. Cicchetti and A. R. Feinstein. High Agreement But Low Kappa: II. Resolving The Paradoxes. *Journal of clinical epidemiology*, 43(6):551–558, 1990. 102

A. Clark, C. Fox, and S. Lappin. *The Handbook Of Computational Linguistics And Natural Language Processing*, volume 57. Wiley. com, 2010. 74

J. Cohen et al. A Coefficient Of Agreement For Nominal Scales. *Educational and psychological measurement*, 20(1):37–46, 1960. 101, 104, 105, 108

M. Collins, L. Ramshaw, J. Hajič, and C. 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. 1, 155

A. J. Conger. Integration And Generalization Of Kappas For Multiple Raters. *Psychological Bulletin*, 88(2):322, 1980. 108

A. Corazza, A. Lavelli, G. Satta, and R. 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)*, 2004. 1, 155

M. Core and J. Allen. Coding Dialogs With The DAMSL Annotation Scheme. In *AAAI fall symposium on communicative action in humans and machines*, pages 28–35. Boston, MA, 1997. 102, 114

R. Craggs and M. Wood. A Two Dimensional Annotation Scheme For Emotion In Dialogue. In *Proceedings of AAAI Spring Symposium: Exploring Attitude and Affect in Text*, 2004. 102

R. Craggs and M. M. Wood. Evaluating Discourse And Dialogue Coding Schemes. *Computational Linguistics*, 31(3):289–296, 2005. 102, 108, 114

R. T. Craig. Generalization Of Scott's Index Of Inter-Coder Agreement. *Public Opinion Quarterly*, 45(2):260–264, 1981. 108

M. Davies and J. L. Fleiss. Measuring Agreement For Multinomial Data. *Biometrics*, pages 1047–1051, 1982. 108

B. Di Eugenio. On The Usage Of Kappa To Evaluate Agreement On Coding Tasks. In *LREC*, 2000. 102, 114

B. Di Eugenio and M. Glass. The Kappa Statistic: A Second Look. *Computational linguistics*, 30(1):95–101, 2004. 102, 103, 114

A. Dubey and F. 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. 1, 155

J. Earley. An Efficient Context-Free Parsing Algorithm. *Communications of the ACM*, 13(2):94–102, 1970. 155, 156

J. C. Earley. *An Efficient Context-Free Parsing Algorithm*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 1968. AAI6907901. 176

K. Erk, A. Kowalski, S. Padó, and M. Pinkal. Towards A Resource For Lexical Semantics: A Large German Corpus With Extensive Semantic Annotation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 537–544. Association for Computational Linguistics, 2003. 104

REFERENCES

R. Facchinetti, M. G. Krug, and F. R. Palmer. *Modality In Contemporary English*. Walter de Gruyter, 2003. 22

A. R. Feinstein and D. V. Cicchetti. High Agreement But Low Kappa: I. The Problems Of Two Paradoxes. *Journal of clinical epidemiology*, 43(6):543–549, 1990. 102

J. L. Fleiss. Measuring Nominal Scale Agreement Among Many Raters. *Psychological bulletin*, 76(5):378, 1971. 105, 108

J. L. Fleiss, B. Levin, and M. C. Paik. *Statistical Methods For Rates And Proportions*. John Wiley & Sons, 2013. 107

R. Garside, G. N. Leech, T. McEnery, et al. *Corpus Annotation: Linguistic Information From Computer Text Corpora*. Longman London, New York, 1997. 66

S. Greenbaum. *Comparing English Worldwide: The International Corpus Of English*. Clarendon Press Oxford, 1996. 66

K. L. Gwet. *Handbook Of Inter-Rater Reliability: The Definitive Guide To Measuring The Extent Of Agreement Among Multiple Raters*. Advanced Analytics Press, 2012. 104, 107

M. A. Haq. *Qawaid-E-Urdu*. Al-Nazer Press, India, 1914. 129

A. Hautli, S. Sulger, and M. Butt. Adding An Annotation Layer To The Hindi/Urdu Treebank. *Linguistic Issues in Language Technology*, 7(1), 2012. 71

A. F. Hayes and K. Krippendorff. Answering The Call For A Standard Reliability Measure For Coding Data. *Communication Methods and Measures*, 1(1):77–89, 2007. 108, 111

A. Heuvelmans and P. Sanders. Beoordelaarsovereenstemming [Inter-Rater Reliability]. *Psychometrie in de praktijk*, pages 443–471, 1993. 108

E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. OntoNotes: The 90% Solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60. Association for Computational Linguistics, 2006. 103, 110

L. Hubert. Kappa Revisited. *Psychological Bulletin*, 84(2):289, 1977. 108

M. Ijaz and S. Hussain. Corpus Based Urdu Lexicon Development. In *the Proceedings of Conference on Language Technology (CLT07), University of Peshawar, Pakistan*, 2007. 73

W. Jiang, H. Xiong, and Q. Liu. Mutipath Shift-Reduce Parsing With Online Training. *CIPS-ParsEval-2009 shared task*, 2009. xxiv, 161

C. R. Johnson, C. J. Fillmore, M. R. Petruck, C. F. Baker, M. Ellsworth, J. Ruppenhofer, and E. J. Wood. Framenet: Theory And Practice, 2002. 105

D. Jurafsky and J. Martin. *Speech And Language Processing: An Introduction To Natural Language Processing, Computational Linguistics, And Speech Recognition*. Prentice Hall series in artificial intelligence. Pearson Prentice Hall, 2009. ISBN 9780131873216. URL `http://books.google.co.uk/books?id=fZmj5UNK8AQC`. xxiv, 158

Y. Kamide, C. Scheepers, and G. T. Altmann. Integration Of Syntactic And Semantic Information In Predictive Processing: Cross-Linguistic Evidence From German And English. *Journal of psycholinguistic research*, 32(1):37–55, 2003. 29

T. A. Khan. *Spatial Expressions And Case In South Asian Languages*. PhD thesis, Bibliothek der Universität Konstanz, 2009. 146, 147, 153

T. A. Khan. Spatial Expressions And Case In South Asian Languages. *Institutional Repository of University of Konstanz (KOPS)*, 2011. 85

K. Krippendorff. Estimating The Reliability, Systematic Error And Random Error Of Interval Data. *Educational and Psychological Measurement*, 30(1):61–70, 1970. 111

K. Krippendorff. Reliability In Content Analysis. *Human Communication Research*, 30(3):411–433, 2004. 2, 3, 101, 103, 104, 108, 111, 114, 123

K. Krippendorff. *Content Analysis: An Introduction To Its Methodology*. Sage, 2012. 101, 104, 108

S. Kulick, R. Gabbard, and M. Marcus. Parsing The Arabic Treebank: Analysis And Improvements. In *Proceedings of the Treebanks and Linguistic Theories Conference*, pages 31–42. Citeseer, 2006. 1, 155

P. Kunitzsch. The Transmission Of Hindu-Arabic Numerals Reconsidered. *The Enterprise of Science in Islam: New Perspectives*, pages 3–21, 2003. 125

J. R. Landis and G. G. Koch. The Measurement Of Observer Agreement For Categorical Data. *biometrics*, pages 159–174, 1977. xx, 104, 107, 108, 109

R. Leblanc and C. N. Fischer. Crafting A Compiler, 1988. 182

G. Leech. The Lancaster Parsed Corpus. *ICAME Journal*, 16:124, 1992. 66

G. Leech. *Adding Linguistic Annotation*. Oxbow Books, 2005. 1, 64

R. J. Light. Measures Of Response Agreement For Qualitative Data: Some Generalizations And Alternatives. *Psychological bulletin*, 76(5):365, 1971. 108

D. M. Magerman and M. P. Marcus. Parsing A Natural Language Using Mutual Information Statistics. In *AAAI*, volume 90, pages 984–989, 1990. 66

M. K. Malik, T. Ahmed, S. Sulger, T. Bögel, A. Gulzar, G. Raza, S. Hussain, and M. Butt. Transliterating Urdu For A Broad-Coverage Urdu/Hindi LFG Grammar. In *LREC*, 2010. 75

C. D. Manning. Part-Of-Speech Tagging From 97% To 100%: Is It Time For Some Linguistics? In *Computational Linguistics and Intelligent Text Processing*, pages 171–189. Springer, 2011. 4

D. Marcu, E. Amorrortu, and M. Romera. Experiments In Constructing A Corpus Of Discourse Trees. In *Proceedings of the ACL99 Workshop on Standards and Tools for Discourse Tagging*, pages 48–57, 1999. 102

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building A Large Annotated Corpus Of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330, 1993. 100

K. Markert and M. Nissim. Towards A Corpus Annotated For Metonymies: The Case Of Location Names. In *LREC*. Citeseer, 2002. 104

P. H. Matthews. *The Concise Oxford Dictionary Of Linguistics*. Oxford University Press, 2007. 6, 8, 9, 11, 19

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. Non-Projective Dependency Parsing Using Spanning Tree Algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics, 2005. 159

M. Meteer, R. Schwartz, and R. Weischedel. Studies In Part Of Speech Labelling. In *Proceedings of the workshop on Speech and Natural Language*, pages 331–336. Association for Computational Linguistics, 1991. 66

M. Mieskes and M. Strube. Part Of Speech Tagging Of Transcribed Speech. In *Proceedings of LREC*, pages 935–938, 2006. 102

M. Mikulová and J. Stepánek. Ways Of Evaluation Of The Annotators In Building The Prague Czech-English Dependency Treebank. In *LREC*, 2010. 101

G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. Introduction To Wordnet: An On-Line Lexical Database. *International Journal of Lexicography*, 3 (4):235–244, 1990. 6, 8, 9, 11, 19

T. Mohanan. *Argument Structure In Hindi*. Stanford: CSLI Publications, 1994. 60, 80, 84, 90, 146, 153

N. Mukhtar, M. A. Khan, and F. T. Zuhra. Probabilistic Context Free Grammar For Urdu. *Linguistic and Literature Review*, 1(1):86–94, 2011. 161

N. Mukhtar, M. A. Khan, and F. T. Zuhra. Algorithm For Developing Urdu Probabilistic Parser. *International journal of Electrical and Computer Sciences*, 12(3): 57–66, 2012a. 161

N. Mukhtar, M. A. Khan, F. T. Zuhra, and N. Chiragh. Implementation Of Urdu Probabilistic Parser. *International Journal of Computational Linguistics (IJCL)*, 3 (1):12–20, 2012b. 159, 161, 162, 193

K. A. Neuendorf. *The Content Analysis Guidebook*. Sage, 2002. 104

M. Niv. Syntactic Disambiguation. *The Penn Review of Linguistics*, 14:120–126, 1991. 66

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. MaltParser: A Language-Independent System For Data-Driven Dependency Parsing. *Natural Language Engineering*, 13(2):95–135, 2007. 159

M. Palmer, D. Gildea, and P. Kingsbury. The Proposition Bank: An Annotated Corpus Of Semantic Roles. *Computational Linguistics*, 31(1):71–106, 2005. 69, 70, 104

M. Palmer, H. T. Dang, and C. Fellbaum. Making Fine-Grained And Coarse-Grained Sense Distinctions, Both Manually And Automatically. *Natural Language Engineering*, 13(2):137–163, 2007. 103, 104, 110

M. Palmer, R. Bhatt, B. Narasimhan, O. Rambow, D. M. Sharma, and F. Xia. Hindi Syntax: Annotating Dependency, Lexical Predicate-Argument Structure, And Phrase Structure. In *The 7th International Conference on Natural Language Processing*, pages 14–17, 2009. 69

R. Passonneau. Computing Reliability For Coreference Annotation. In *Proceedings of the Language Resources and Evaluation Conference (LREC 2004)*, 2004. 103, 114

R. J. Passonneau and D. J. Litman. Empirical Analysis Of Three Dimensions Of Spoken Discourse: Segmentation, Coherence, And Linguistic Devices. *NATO ASI SERIES F COMPUTER AND SYSTEMS SCIENCES*, 151:161–194, 1996. 114

K. Pearson, A. Lee, E. Warren, A. Fry, and C. D. Fawcett. Mathematical Contributions To The Theory Of Evolution. IX.--On The Principle Of Homotyposis And Its Relation To Heredity, To The Variability Of The Individual, And To That Of The Race. Part I.--Homotyposis In The Vegetable Kingdom. *Proceedings of the Royal Society of London*, 68(442-450):1–5, 1901. 109

F. Pereira and Y. Schabes. Inside-Outside Reestimation From Partially Bracketed Corpora. In *Proceedings of the 30th annual meeting on Association for Computational Linguistics*, pages 128–135. Association for Computational Linguistics, 1992. 66

M. Poesio and R. Vieira. A Corpus-Based Investigation Of Definite Description Use. *Computational linguistics*, 24(2):183–216, 1998. 102, 114

R. Popping. Overeenstemmingsmaten Voor Nominale Data. *Unpublished Ph. D.-thesis, University of Groningen*, 1983. 108

J. J. Randolph, A. Thanks, R. Bednarik, and N. Myller. Free-Marginal Multirater Kappa (Multirater κfree): An Alternative To Fleiss Fixed-Marginal Multirater Kappa. In *Joensuu learning and instruction symposium*. Citeseer, 2005. 104

G. Raza. Inferring Subcat Frames Of Verbs In Urdu. In *LREC*, 2010. 26

G. Raza. Subcategorization Acquisition And Classes Of Predication In Urdu. *Institutional Repository of University of Konstanz (KOPS)*, 2011. 19, 20, 24, 84, 127, 130, 131, 147

D. Reidsma and J. Carletta. Reliability Measurement Without Limits. *Computational Linguistics*, 34(3):319–326, 2008. 154

S. M. J. Rizvi. Indications Of Urdu Tetravalent Verbs Having 'Oblique Agents' In The Argument Structure. In *Proceedings of the LFG08 Conference, CA: CSLI Publications*, volume 19, 2008. 38, 146

P. Saint-Dizier. Syntactic And Semantic Frames In PrepNet. In *IJCNLP*, pages 763–768, 2008. 14

G. Sampson. Briefly Noted-English For The Computer: The SUSANNE Corpus And Analytic Scheme. *Computational Linguistics*, 28(1):102–103, 2002. 66

B. Santorini. Part-Of-Speech Tagging Guidelines For The Penn Treebank Project (3rd Revision). *Institutional Repository of University of Pennsylvania*, 1990. 66

A. Schiller, S. Teufel, and C. Thielen. Guidelines Für Das Tagging Deutscher Textcorpora Mit STTS. *Manuscript, Universities of Stuttgart and Tübingen*, 1995. 66

R. L. Schmidt. *Urdu: An Essential Grammar*. Routledge, 2013. 6, 20, 23, 28, 30, 56, 78, 86, 129

W. A. Scott. Reliability Of Content Analysis: The Case Of Nominal Scale Coding. *Public opinion quarterly*, 1955. 101, 105

A. L. Siddiqui. *Urdu Lughat: Tarikhi Usul Par*, volume 1. Taraqqi-Urdu Borad, Pakistan, 1977. 129

S. Siegel and N. J. Castellan. Nonparametric Statistics For The Behavioral Sciences. *McGraw-HiU Book Company, New York*, 1988. 102

S. Siegel and N. Castellan Jr. Jr Nonparametric Statistics For The Behavioral Sciences (2e) McGraw-Hill. *New York*, 1988. 104

J. Sim and C. C. Wright. The Kappa Statistic In Reliability Studies: Use, Interpretation, And Sample Size Requirements. *Physical therapy*, 85(3):257–268, 2005. 108

W. Skut, B. Krenn, T. Brants, and H. Uszkoreit. An Annotation Scheme For Free Word Order Languages. In *Proceedings of the fifth conference on Applied natural language processing*, pages 88–95. Association for Computational Linguistics, 1997. 68, 74

A. Stevenson. *Oxford Dictionary Of English*. Oxford University Press, 2010. 6, 8, 9, 11, 19

M. Stevenson and R. Gaizauskas. Experiments On Sentence Boundary Detection. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, pages 84–89. Association for Computational Linguistics, 2000. 102

A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. Van Ess-Dykema, and M. Meteer. Dialogue Act Modeling For Automatic Tagging And Recognition Of Conversational Speech. *Computational linguistics*, 26 (3):339–373, 2000. 102, 114

S. Sulger, M. Butt, T. H. King, P. Meurer, T. Laczkó, G. Rákosi, C. B. Dione, H. Dyvik, V. Rosén, K. De Smedt, et al. ParGramBank: The ParGram Parallel Treebank. *Association for Computational Linguistics*, 2013. 70

L. Tesnière and J. Fourquet. *Eléments De Syntaxe Structurale*, volume 1965. Klincksieck Paris, 1959. 65

B. N. Tiwari. *Hindi Basha*. KitAb Mahal, Allahabad, 2004. ISBN 81-225-0017-X. 60, 86

R. Tsarfaty and K. Sima'an. Three-Dimensional Parametrization For Parsing Morphologically Rich Languages. In *Proceedings of the 10th International Conference on Parsing Technologies*, pages 156–167. Association for Computational Linguistics, 2007. 1, 155

R. Tsarfaty, D. Seddah, Y. Goldberg, S. Kuebler, M. Candito, J. Foster, Y. Versley, I. Rehbein, and L. 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. 1, 155, 162

R. Tsarfaty, D. Seddah, S. Kübler, and J. Nivre. Parsing Morphologically Rich Languages: Introduction To The Special Issue. *Computational Linguistics*, 39(1):15–22, 2013. 1, 4, 155, 156, 160, 162, 166

S. Urooj, S. Hussain, F. Adeeba, F. Jabeen, and R. Parveen. CLE Urdu Digest Corpus. *LANGUAGE & TECHNOLOGY*, page 47, 2012. 73

N. Veilleux and M. Ostendorf. Probabilistic Parse Scoring With Prosodic Information. In *Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on*, volume 2, pages 51–54. IEEE, 1993. 66

J. Véronis. A Study Of Polysemy Judgements And Inter-Annotator Agreement. In *Programme and advanced papers of the Senseval workshop*, pages 2–4, 1998. 102

S. Wallis. *Searching Treebanks And Other Structured Corpora*. Mouton de Gruyter, 2008. 66

M. J. Warrens. Inequalities Between Multi-Rater Kappas. *Advances in data analysis and classification*, 4(4):271–286, 2010. 104, 108

R. M. Weischedel, D. M. Ayuso, R. J. Bobrow, S. Boisen, R. Ingria, and J. Palmucci. Partial Parsing: A Report On Work In Progress. In *HLT*, 1991. 66

R. Zwick. Another Look At Interrater Agreement. *Psychological Bulletin*, 103(3):374, 1988. 102

Declaration

I herewith declare that I have produced this thesis without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This thesis has not previously been presented in identical or similar form to any other German or foreign examination board.

The thesis work was conducted from 2011 to 2014 under the supervision of Prof. Dr. Miriam Butt at Fachbereich Sprachwissenschaft in Universität Konstanz, Germany.

KONSTANZ, GERMANY