Urdu Dependency Parser: A Data-Driven approach

Wajid Ali, Sarmad Hussain
National University of Computer Sciences (NUCES), Lahore, Pakistan; Center for Language Engineering Al-Khawarizmi Institute of Computer Sciences, University of Engineering and Technology, Lahore, Pakistan
wajid.ali@msn.com, sarmad.hussain@kics.edu.pk

Abstract

In this paper, we present what we believe to be the first data-driven dependency parser for Urdu. The parser was trained and tuned using MaltParser system, a system for data-driven dependency parsing. The Urdu dependency treebank (UDT) is used for training and testing of the Urdu dependency parser, is also presented first time. The UDT contains corpus of 2853 sentences which are annotated at multiple levels such as part-of-speech (POS) level, chunk (phrase level) and dependency relations level. The UDT also contains information about the token counter, head of current token. The annotation is done manually to build UDT. Urdu Dependency Parsing system is evaluated by conducting a series of experiments. All experiments are performed using Maltparser default algorithm with different feature models. Initial, a base line simple feature model consisting word position, word, head and dependency relation is used for Urdu dependency parsing. Then feature model is enhanced by adding part-of-speech (POS) and chunk (Phrase level) information. The results of all parsing experiments are reported. The overall best labeled accuracy (LA) achieved 74.48% and 90.14% of unlabeled attachment score (UAS) is achieved. The error analysis is performed by comparing output data with treebank test data which manual parsed to analyze and classify the different types of errors produced by the parser. This is very useful to identify the future directions for future expansion of the treebank and for improving the parsing accuracy.

1. Introduction

Parsing is a process of assigning a syntactic representation to a natural language sentence and analyzing the grammatical structure of a natural language sentence. It is one of the major tasks and core component in many systems for natural language processing which help in understanding the natural language. It is useful in several natural language processing (NLP) applications like machine translation, word sense disambiguation, question answering, summarization and natural language text understanding etc. There are two structure formalisms such as a phrase structure (PS) or dependency structure (DS) formalism is used for assigning the syntactic representation to a natural language sentence through parsing. Parsing a sentence according to phrase structure formalism would essentially output a tree with phrase structure information. This formalism has been very popular for fixed order languages especially English language.

The dependency structure formalism using dependency grammar presented by Tesni`ere (1959) has been attracting many adherent, mainly because of its ability to handle free word order languages. It has been suggested that free word order languages can be handled better using the dependency based framework than the constituency based one [12], [15], [4], [1]. The basic difference between a constituent (phrase structure) based representation and a dependency structure based representation is the lack of non-terminal nodes in the latter. Parsing a sentence according to the dependency structure formalism indicates the dependency relations between the words in a parse tree. A dependency is a relation between two words, one of them being a head or regent and the other a dependent. There are two approaches used for dependency parsing, data-driven dependency parsing and grammar driven dependency parsing approach [5]. In this paper, we focus on data-driven dependency parsing approach only. Data driven parsers need large amount of manually annotated parsed data which is called a Treebank. Such data for Urdu was not available and developed during this research first time.

Dependency Grammar and Parsing is briefly overviewed in Section 2. Urdu Dependency Treebank is presented in Section 3. Section 4 describes the Maltparser. Related work on dependency parsing for free word order languages is presented in Section 5. Methodology discussed in Section 6. In Section 7 we give the results and discussion. We conclude our paper in Section 8.

2. Dependency Grammar and Parsing

2.1. Dependency grammars

French linguist Tesni`ere (1959) has created dependency grammar which is usually taken as the starting point of the modern theoretical tradition of dependency grammar. Dependency grammar (DG) is a set of rules that describes the dependencies. Dependency is an asymmetrical relation between a head or regent and a dependent. Heads and dependents are related immediately (e.g. there are no non-terminals). The observation that drives dependency grammar is simple: every dependent (word or phrase) depends on another head, except head of the sentence, which is the root[1] of the sentence.

Among the more well-known theories of dependency grammar, besides the theory of structural syntax developed by Tesni`ere, we find Word Grammar (WG) proposed by Hudson [12][13], Functional Generative Description (FGD) [11], Dependency Unification Grammar (DUG) [10], Meaning-Text Theory (MTT) [4][16]. In addition, constraint-based theories of dependency grammars have a strong tradition, represented by Constraint Dependency Grammar (CDG) by Menzel and Schr¨oder, 1998) [9], Weighted Constraint Dependency Grammar (WCDG) (Schr¨oder, 2002), Functional Dependency Grammar (FDG) [18], Constraint Grammar (CG) (Karlsson, 1990; Karlsson et al., 1995), Topological Dependency Grammar (TDG) [3], Extensible Dependency Grammar (XDG).

2.2. Dependency parsing

The fundamental idea of dependency parsing is that the parsing crucially involves establishing relations between words in the sentence. This is illustrated in figure 1, which depicts the analysis of a short sentence taken from the Wall Street Journal section of the Penn Treebank [8].

[1] The root is alternatively termed as head or central element

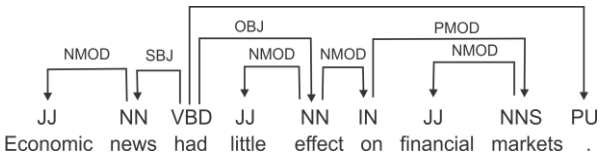

Figure 1: Dependency structure of English sentence

In this example, the syntactic structure is built up by recognizing a subject relation (SBJ) from the finite verb *had* to the noun *news*, a nominal modifier relation (NMOD) from *news* to the adjective *Economic*, an object relation (OBJ) from *had* to the noun *effect*, and so on [8]. Dependency parsing can be broadly divided into grammar-driven dependency parsing and data-driven dependency parsing [5]. We briefly discuss both parsing approaches in subsections.

2.2.1. Grammar-driven parsing. In the grammar-driven approach, parsing is modeled by the abstract problem of grammar parsing [14]. A grammar parsing algorithm is then used to compute the analyses of a given input string. The grammar may be hand-crafted or it may be wholly or partially induced from corpus data. There are two type of grammar-driven parsing [5], (i) a context free grammar and (ii) constraints based grammar parsing. Most of the modern grammar-driven dependency parsers parse by using constraints based grammars. They view parsing as a constraint-satisfaction problem and used constraint based grammar for parsing.

2.2.2. Data-driven parsing. In the data-driven approach to parse, a formal grammar is no longer a necessary component of the parsing system [14]. Instead, the mapping from input strings to output analyses is defined by an inductive mechanism applying to a text sample from the language to be analyzed. As no rules are required, corpus is used in data-driven parsing, mostly treebank is used. There are also two approaches of modeling a data-driven parsing [5], i) Graph based models, ii) Transition based models. MaltParser used transition based approach.

3. Urdu Dependency Treebank

Urdu dependency treebank (UDT) was developed using a multi-level and multi-representational annotation such as part-of-speech, chunking and dependency relation representation. UDT contains 2853 sentences with an average length of 14.03 words and has 40012 tokens. The UDT texts were chosen from large corpus containing all type of sentences, simple or complex. In corpus, small size to large size

sentences available which are simple or complex. As this is initial effort, so we sort the sentences on the bases on number of word in it to remove the large sentences. However, texts including complex ambiguities were also avoided as such as possible, being removed from the corpus. All punctuation marks are also ignored. The corpus is manually annotated using a POS tagset which contains 35 POS tags [22], [21], 9 chunk (Phrase Level) tags [21] and small set of dependency relations [21]. As this initial effort and the dependency annotation scheme is still undergoing process, only six dependency relations tags are used for UDT which are, subject (subj), object (obj), secondary object (obj2) , adjectival modification (jmod), adverbial modification (rmod) and noun modification (nmod).

The treebank data is converted to CoNLL data format[2] for MaltParser parsing system. The CoNLL fields which we used for feature models are: ID (Token counter, starting at 1 for each new sentence), FORM (Word form), CPOSTAG (Coarse-grained part-of-speech tag), POSTAG (Fine-grained POS tag), HEAD (Head of the current token, which is a value of ID). We use zero (0) for Head of the sentence and value of ID for phrases head, DEPREL (Dependency relation to the HEAD). The CoNLL format and annotation (token counter, word form, POS tag, chunk tag, head information and dependency relation) is demonstrated by this typical sentence example (1).

(1) علی نے بازار میں ایک کالی گائے دیکھی۔

Ali ne baazaar mein aik kali gay daikhi
Ali CM market in a black cow see-past
"Ali saw a black cow in the market."

ID	FORM	POSTAG	CPOSTAG	HEAD	DEPREL
1	علی	NNP	B-NP	8	subj
2	نے	PP	I-NP	8	subj
3	بازار	NN	B-NP	8	obj2
4	میں	PP	I-NP	8	obj2
5	ایک	CD	B-NP	7	nmod
6	کالی	JJ	B-JJP	7	jmod
7	گائے	NN	B-NP	8	obj
8	دیکھی	VB	B-VP	0	main

[2] CoNLL format is a text format for storing sentence structure, like Subject, Predicate, direct Object. It's not XML and stored in UTF-8 encoding. It is standard format and used by many parser.

4. Maltparser

Transition-based approach is used by MaltParser for dependency parsing. MaltParser has two essential components [5]:

- A transition system for mapping sentences to dependency trees.
- A classifier for predicting the next transition for every possible system configuration.

Dependency parsing can be realized as deterministic search through the transition system, guided by the classifier using two components. With this technique, parsing can be performed in linear time for projective dependency trees and quadratic time for arbitrary (possibly non-projective) trees [5].

4.1. Transition System

There are built-in transition systems in MaltParser but we limit our attention to the system that has been used in the parsing experiments: the arc-eager projective system first described in Nivre [6]. A configuration in the arc-eager projective system contains a stack holding partially processed tokens, an input buffer containing the remaining tokens, and a set of arcs representing the partially built dependency tree. There are four possible transitions (where top is the token on top of the stack and next is the next token in the input buffer):

- LEFT-ARC(r): Add an arc labeled r from next to top; pop the stack.
- RIGHT-ARC(r): Add an arc labeled r from top to next; push next onto the stack.
- REDUCE: Pop the stack.
- SHIFT: Push next onto the stack.

Although this system can only derive projective dependency trees, the fact that the trees are labeled allows non-projective dependencies to be captured using the pseudo-projective parsing technique.

4.2. Classifiers

Classifiers can be induced from treebank data using a wide variety of different machine learning methods. The task of the classifier is to map a high-dimensional feature vector representation of a parser configuration to the optimal transition out of that configuration. The features used in our system all represent attributes of tokens and have been extracted from the fields of the

CoNLL data representation as discussed above in section 3.

5. Related work to dependency parsing

The related work to dependency parsing using MaltParser is presented in this section. The statistical approaches for Japanese dependency analysis are based on a probabilistic model consisting two steps [19] [20]. First, they estimate modification probabilities. Secondly, they searched the optimal combination of dependencies from the all candidates' dependencies. The maximum sentence accuracy was achieved 42.94% and dependency relations accuracy was 87.01%. Another approach of dependency parsing for Japanese dependency analysis was performed using cascading chunking [19]. The dependency relations accuracy improved to 89.29% with 47.53% sentence accuracy. A deterministic dependency parser based on memory-based learning is presented and dependency analysis of Swedish was performed using MaltParser [7]. A label accuracy score (LAS) of 80.6% was achieved.

A statistical dependency parser was presented for four languages (Arabic, Bulgarian, Italian and Slovene) [22]. He used MaltParser for training and parsing experiments. For Arabic language, Prague Arabic Dependency Treebank is used which had training (1,460 sentences; 54,379 tokens) and test (146 sentences; 5,373 tokens) data. The results vary from 50.7% to 66.9% labels accuracy. The BulTreeBank containing training (10,911 sentences; 159,395 tokens) and test (2,310 sentences; 36,756 tokens) data is used for Bulgarian language. The label accuracy was achieved 67.6%. Italian was parsed trained and test on Turin University Treebank (training data: 1,500 sentences (44,616 tokens) and tested on 150 sentences (4,172 tokens)). The reported accuracy is 81.8%. The Slovene Dependency Treebank contained a training set (1,534 sentences, 28,750 words) and a test set (402 sentences, 6,390 words) for experiments. Results for Slovene varied from 50.7% to 73.4% labeled accuracy. A dependency parser was developed for Thai [17]. It is part of an ongoing project in developing syntactically annotated Thai corpus. The corpus consists of 2692 sentences. They achieved 83.64% as the root accuracy, 78.54% as the dependency accuracy and 53.90% as the complete sentence accuracy.

Two stage constraint based hybrid approach was presented for Indian languages dependency parsing [2]. They define the two stages and show how different grammatical construct are parsed at appropriate stages. The best labeled attachment and labeled accuracies are 68% and 70.6%. Some improvement was made by working on semantics features [] and the labeled attachment accuracy was improved to 69.47% and labeled accuracies are 71.71%.

6. Methodology

Architecture and computational model of Urdu dependency system is presented in the subsections.

6.1. Architecture

The system architecture of data-driven dependency parser for Urdu is presented in Figure 2 below. An Urdu dependency treebank sentence is input of the Urdu dependency parsing system which used MaltParser system. The input data is prepared in a in the form of CoNLL data format. MaltParser is trained on UDT and parse the given data to mark dependency relations with heads information.

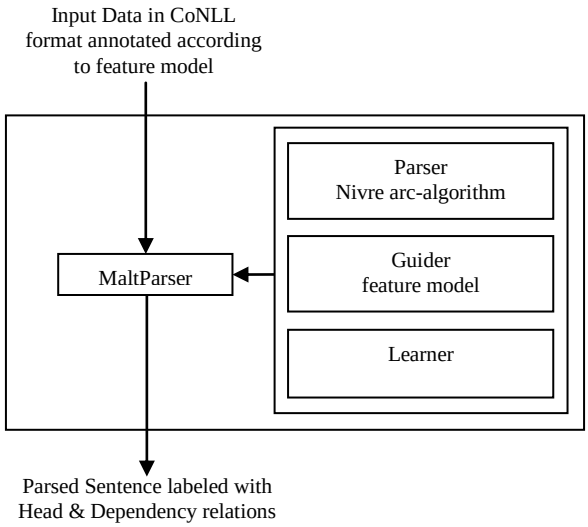

Figure 2: Urdu Dependency Parser Architecture

6.2. Experiments

Urdu dependency parsing system is evaluated by a series of experiments. The treebank data is trained and tested using MaltParser. Nivre arc-eager algorithm is used for parsing in training and testing but with different feature models. Six different feature models are used. In initial, a simple feature model containing token counter, word form, head of current token and dependency relation of current token is used. The feature model is extended by adding part-of-speech and

chunk tags. So, total number of six experiments are performed which are listed in table 1.

Table 1: Experiments

No.	Parsing Algorithm	Features Model
1.	Nivre arc-eager	ID, FORM, HEAD, DEPREL
2.	Nivre arc-eager	ID, FORM, POSTAG, HEAD, DEPREL
3.	Nivre arc-eager	ID, FORM, CPOSTAG, HEAD, DEPREL
4.	Nivre arc-eager	ID, FORM, CPOSTAG (IOB-Chunk tag), HEAD, DEPREL
5.	Nivre arc-eager	ID, FORM, POSTAG, CPOSTAG, HEAD, DEPREL
6.	Nivre arc-eager	ID, FORM, POSTAG, CPOSTAG (IOB-Chunk tag), HEAD, DEPREL

7. Results and Discussion

7.1. Results

MaltParser was trained and tested on an Urdu Dependency Treebank data which contains 2853 sentences with an average length of 14.03 words and has 40012 tokens. The training data has 2187 sentences and test data has 666 sentences. The experiments were performed. The evaluation metric for MaltParser is, unlabeled attachment score (UAS), i.e., the percentage of tokens with correct HEAD and the label accuracy (LA), i.e., the percentage of tokens with correct DEPREL. The results are given below in table 2.

Table 2: Experiment Results

No.	Experiments	Evaluation	
		UAS	LA
1.	Nivre arc-eager with feature model (ID, FORM, HEAD, DEPREL)	67.59	55.86
2.	Nivre arc-eager with feature model (ID, FORM, POSTAG, HEAD, DEPREL)	70.52	58.28
3.	Nivre arc-eager with feature model (ID, FORM, CPOSTAG, HEAD, DEPREL)	73.29	60.57
4.	Nivre arc-eager with feature model (ID, FORM, CPOSTAG (IOB-Chunk tag), HEAD, DEPREL)	76.01	62.82
5.	Nivre arc-eager with feature model (ID, FORM, POSTAG, CPOSTAG, HEAD, DEPREL)	86.91	72.19
6.	Nivre arc-eager with feature model (ID, FORM, POSTAG, CPOSTAG (IOB-Chunk tag), HEAD, DEPREL)	90.14	74.48

7.2. Discussion

Urdu dependency parsing system is trained and tested using MaltParser and UDR. Parsing using only dependency relation and head information leads to very low accuracy because of unseen word. The parser ambiguity is over come by adding the information such as part-of-speech and chunking which helps to improvement the accuracy of correct dependency relation labeling and head identification. It is also observed chunk tags with IOB boundary (IOB-Chunk) give much better results as compare to use only chunk tags. There were also cases where the parser failed to mark any dependency relation for a token within a chunk. This happened because the token within the chunk failed to satisfy all the required information. It is also observed that the incorrect head information leads to incorrect dependency relation. So, head information also plays a vital role in correction dependency label. There are total 2662 relations in testing data which contains 666 sentences. There are total 679 dependency relations which are marked incorrect or fail to mark by MaltParser.

The error analysis is performed by comparing output data with manual parsed data to analyze and classify the different types of errors produced by the parser. This is very useful to identify the future directions for future expansion of the treebank and for improving the parsing accuracy. Below, we show the error analysis for incorrect labels marked for relations subject, object, object2, jmod, nmod and heads.

7.2.1. Subject Error. Subject conflicts with Object for nominative case (with zero case markers) and ‘ko’ post positions primarily. This happened because it is difficult to distinguish between the semantics of a subject chunk with that of an Object chunk. The errors of subject are as following:

Table 3: Subject Error

Conflicts with	Object	Object2	jmod	nmod
Subject	316	29	24	19

During manual annotation of treebank, it was observed that a large number of instances of subject have a zero post position i.e. no post position and most ko instance marked object relation. These errors can be minimized using morphological information.

7.2.2. Object Error. As mentioned in above subject and Object have a high degree of conflicts. Similarly, the relation object2 is also difficult to disambiguate from object. It is also observed that when adjective appear as an object relation in the sentence, the parser was unable to label it correctly.

Table 4: Object Error

Conflicts with	Subject	Object2	jmod	nmod
Object	316	151	70	16

As there are not as much object2 relations in the corpus as subject relation. The morphological information and some constraint grammar based rules can help to minimize the error.

7.2.3. Object2 Error. In the case of Object2, parser records the highest number of conflicts with Object. The reason for conflict with object is also the same as above case markers conflict.

Table 5: Object2 Error

Conflicts with	Subject	Object	jmod	nmod
Object2	29	151	9	4

The numbers of errors when the parser failed to label any dependency relation are 41.

7.2.4. Heads Error. The error analysis, as described above, was done for incorrect and unmarked dependency labels. We now describe the error analysis for the head attachment of these dependency labels because dependency relations accuracy also depends on Head accuracy. Again, the results for checking accuracy of dependency labels marked along with correct attachment with their corresponding parent are lower. For relations like jmod and nmod whose parent chunk is generally not a verb group could also be the reason for lower accuracy.

8. Conclusion

This paper presents a data-driven dependency parsing approach to parse Urdu sentences. An Urdu dependency parsing system is trained and tested with MaltParser system. Urdu dependency treebank represents the data for training and testing is used in all experiments. Nivre arc-agear algorithm is used in parsing system to train and test the data with different feature models. A series of experiments are conducted using six feature models to increase the accuracy of Urdu dependency parsing system. The parser achieved maximum unlabeled attachment score (UAS) of 90.14% and label attachment of 74.48%.

The results obtained in this research can be considered promising, entailing a continuation on the same track and leaving a lot of space for improvement. The Urdu dependency system is based on data-driven dependency parsing and MaltParser is also a data driven system, it requires a relatively large amount of training data. The parser trained in this research can be used in order to process new Urdu text which, after correction by a human expert, could be used to create a larger dataset. However, a new set of annotation guidelines together with extended and refined morpho-syntactic and syntactico-semantic information especially morphological information and dependency annotation scheme needs to expand the current treebank. Punctuation marks should also be taken into account in a future extended version of the current treebank. Since the error analysis showed that case markers and some lexical properties have a rather high error rate, this needs to be considered too before annotating new material. Furthermore, the selection of the texts should be looser, allowing at least complex sentences.

9. References

[1] A. Bharati, V. Chanitanya and R. Sangal, Natural Language Processing: A Paninian Perspective, Prentice-Hall of India, New Delhi, 1994.

[2] A. Bharati, S. Husain, D. M. Sharma and R. Sangal, "A Two-Stage Constraint Based Dependency Parser for Free Word Order Languages", in proc. *of the COLIPS International Conference on Asian Language Processing 2008 (IALP)*, Chiang Mai, Thailand, 2008.

[3] D. Duchier and R. Debusmann, "Topological dependency trees: A constraint-based account of linear precedence," in proc. *of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2001, pp. 180–187.

[4] I. A. Mel'cuk, *Dependency Syntax: Theory and Practice*, State University, Press of New York, 1988.

[5] J. Nivre, *Inductive Dependency Parsing*, Springer, 2006.

[6] J. Nivre, J. Hall and J. Nilsson, "MaltParser: A Data-Driven Parser-Generator for Dependency Parsing," in proc. *of the fifth international conference on Language Resources and Evaluation (LREC2006)*, May 24-26, 2006, Genoa, Italy, pp. 2216-2219

[7] J. Nivre, , J. Hall and J. Nilsson, "Memory-based dependency parsing," in proc. *of the 8th Conference on Computational Natural Language Learning (CoNLL)*, Boston, MA, 2004 pp. 49–56.

[8] M. Marcus, B. Santorini, and M.A. Marcinkiewicz, "Building a large annotated corpus of English: The Penn Treebank", Computational Linguistics 1993.

[9] P. M. Harper and A. R. Helzerman, "Extensions to constraint dependency parsing for spoken language processing", Computer Speech and Language, 1995 pp.187–234.

[10] P. Hellwig, "Dependency unification grammar," in proc. of the 11th conference on Computational linguistics, COLING 1986, Bonn, Germany, 1986, pp. 195–199.

[11] P. Sgall, E. Haji˘cova and J. Panevova, The meaning of the sentence in its semantic and pragmatic aspects, Dordrecht: D. Reidel, 1986.

[12] R. A. Hudson, *Word Grammar*, Blackwell, 1984.

[13] R. A. Hudson, *English Word Grammar*, Blackwell, 1990.

[14] S. Kübler, R. McDonald, and J. Nivre, *Dependency Parsing*, Morgan & Claypool Publishers series, 2009.

[15] S. M. Shieber, "Evidence against the context freeness of natural language," In Linguistics and Philosophy, 1985, p. 8, 334–343.

[16] S. Starosta, *The Case for Lexicase: An Outline of Lexicase Grammatical Theory*, Pinter Publishers, 1988.

[17] S. Tongchim, R. Altmeyer, V. Sornlertlamvanich and H. Isahara, "A Dependency Parser for Thai", Proceedings of The sixth international conference on Language Resources and Evaluation, Marrakech, Morocco, May 28-30, 2008.

[18] T. J¨arvinen and P. Tapanainen, (1998). Towards an implementable dependency grammar. In Kahane, S. and Polgu`ere, A. (eds), Proceedings of the Workshop on Processing of Dependency-Based Grammars, pp. 1–10.

[19] T. Kudo and Y. Matsumoto, "Japanese dependency analysis using cascaded chunking," *In Sixth Conference on Natural Language Learning*, Taipei, Taiwan, 2002.

[20] T. Kudo and Y. Matsumoto, "Japanese dependency analysis based on support vector machines," in Joint Sigdat *Conference On Empirical Methods In Natural Language Processing and Very Large Corpora*, R. Nicole,

[21] W. Ali, *Data-Driven Dependency Parsing for Urdu*, MS (MPhil), Computer Sciences thesis, Department of Computer Sciences, National University of Computer and Emerging Sciences (NUCES), Lahore, Pakistan., un published

[22] W. Ali and S. Hussain, "A hybrid approach to Urdu Verb Phrase chunking", in proc. *The workshop on Asian Language Resources (ALR-8), COLING 2010*, Beijing, China, 21-28 August 2010, pp. 137–143.