Dependency Parsing using the URDU.KON-TB Treebank

Saima Munir
Department of Computer Science
& Information Technology
University of Sargodha, 40100
Sargodha, Pakistan

Qaisar Abbas, PhD
Department of Computer Science
& Information Technology
University of Sargodha, 40100
Sargodha, Pakistan

Bushra Jamil
Department of Computer Science
& Information Technology
University of Sargodha, 40100
Sargodha, Pakistan

ABSTRACT
In this paper, we present evaluation of URDU.KON-TB in the dependency parsing domain. The URDU.KON-TB treebank is developed on the bases of the phrase structure and hyper dependency structure which are only functional constituent's label. Treebank was annotated with three levels of annotation tagset, the semi-semantic POS (SSP), semi-semantic Syntactic (SSS) and Functional (F) tagset and was checked for the Phrase Structure Parsing domain. To evaluate this treebank in the Dependency Parsing domain we have selected MaltParser. To use data in the parser, we have converted the URDU.KON-TB treebank annotated data according to the CONLL format. The compatibility of data to CoNLL is also measured along with usability of data in the dependency parsing domain. To make the data compatible, few assumptions are taken. The converted data is used to evaluate the system by dividing 80% data as training data and 20% data as testing data. We have performed eight experiments. Four experiments are conducted with six different feature models with converted data. The experiments results show URDU.KON-TB treebank is not suitable for the dependency parsing as dependency relation because Head information was missing in the treebank. We then performed four experiments with an assumption based enhancement by adding Head information. The algorithm used to train and test data is Nivre arc-agear algorithm. The new experiments show this treebank data can be used to develop new dependency treebank for Urdu.

General Terms
URDU.KON-TB, MaltParser and Dependency parsing

Keywords
Phrase structure parsing, Data Driven Dependency Parsing, MaltParser

1. INTRODUCTION
Urdu is a South Asian and the national language of Pakistan with rich morphology [1]. It is an Indo-Aryan language, and is a free phrase-order language. In a free phrase language, the phrases within a sentence have free order but the words within a phrase have a fixed order [3]. In parsing we assign a syntactic representation and analyze the grammatical structure of a natural language sentence [2]. The parsing is done using a treebank data. A treebank is a text corpus of sentences annotated with syntactic structure. Corpus annotation is the process of adding the interpretative linguistic information in form of the labels or tags of text. Tag is identifying the class of words as part of speech (POS) and arrangement of words and phrases in form of sentence is a syntactic tagging [1]. The most popular parsing representations are the phrase structure (PS) and the dependency structure (DS) [2].

For a treebank, the traditional annotation structure is Phrase structure. In Phrase Structure node represents the noun phrase NP and a verb phrase VP. Example of such a sentence is shown in Figure 1.

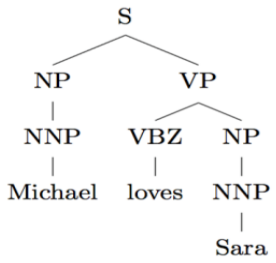

Fig 1: Phrase structure example

The set of rules used for describing dependencies is a Dependency grammar. The asymmetrical relation between a head and dependent is a Dependency [2]. Head and Dependent are related. Every dependent (word or phrase) depends on head of the sentence is a main verb [2]. In dependency parsing the relations between words of the sentence are established [2]. Parsing can be divided into grammar-driven dependency parsing and data-driven dependency parsing. Example of dependency structure is shown in Figure 2.

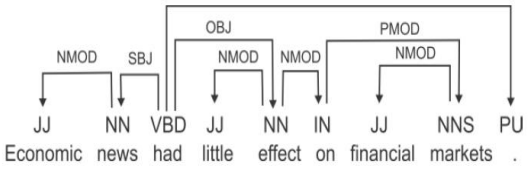

Fig 2: Dependency Structure of an English Sentence

In this example, the dependency structure is built up with 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 head to the noun effect, and so on [2]. In data-driven dependency parsing we map input strings to output using inductive mechanism. This mechanism is applied to a text sample taken from the language [3]. In Grammar driven dependency parsing approach, we use grammar parsing algorithm for computing the analysis of a given selected input string [3]. your own material.

2. URDU.KON-TB Treebank
In this section, we discuss bracketing notation of treebank, CFG and URDU.KON-TB. The Bracketing is computational treebank as shown in Figure 3. S in this bracketed form is the same two child nodes NP and VP as given in the tree representation. Similarly, the VP is the same structure of VBZ and NP at parallel level and so on [1]. So, URDU.KON-TB treebank is developed along with POS and syntactic an-notation [1].

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

Fig 3: Bracketing notation of Treebank

The context free grammar (CFG) and constraints based grammar parsing are the types of grammar-driven parsing. The first one was performed on the URDU.KON-TB treebank earlier by Abbas [1]. URDU.KON-TB treebank adopted the same phrase structure annotation and hyper dependency structures. It is performed in three steps is as following: The collection of sentences in the form of a corpus, Manufacturing of an annotation scheme and the employment of the annotation scheme on the said corpus [1].
The corpus contains 1400 sentences, in URDU.KON-TB treebank are divided into 80% training data and 20% test data. It describes twenty-two semi-semantic part of speech tagset, twenty-six semi-semantic syntactic tagset and eighteen functional tagset. A context free grammar is extracted from this training data. Urdu parser is an extended version of dynamic programming algorithm [1]. Earlier developed parsers view parsing as a constraint-satisfaction problem and for parsing them use constraint based grammar [2]. The MaltParser is also known to have good result on dependency parsing discussed next section. To parse UDU.KON-TB treebank in the Dependency Parsing domain, the treebank already contains the POS and syntactic tagging (chunk/phrase level annotation) annotation except the last one option of dependency relations which are missing in the URDU.KON-TB treebank. But it has functional tagset.

3. MALTPARSER

Maltparser is a system based on data-driven dependency parsing. It allows user-defined feature models that contain lexical features, part-of-speech features and dependency feature. Three components of parsing methodology are the deterministic parsing algorithms for building labeled dependency graphs, History-based models for predicting the next parser action at nondeterministic choice points and the discriminative learning to map histories to parser actions. MaltParser achieve state-of-the-art accuracy for languages with short distance relation. MaltParser system use Nivre arc-agear algorithm in parsing system to train and test the data [3]. It uses four transitions first is the shift for push the next word in the buffer onto the stack, second is Left-Arc for add an arc from the topmost word on the stack. Third is Right-Arc to add an arc from the second-topmost word on the stack and four is the Reduction for pop the stack [2], [7], [14], [18], [19]. Classifier induces from treebank data using different machine learning methods with predict next action based on the feature vector. Feature vector is a setting in which change to suit the data with different features of the data like POS tags, words etc. Our system uses all features that represent all attributes of tokens. These features have been extracted from the fields of the CoNLL data representation [2]. MaltParser for a new language for need to be optimize Parsing algorithm, Feature model and Learning algorithm [2],[9],[18]. Moreover, the MaltParser is popular for its dependency parsing. So, we have a cushion to work and have a motivation to produce quite good results in this untouched domain. To prove our argument, the literature is presented next section to fulfill the evidence.

4. LITERATURE REVIEW

In 2010, Ali et al. worked on data-driven dependency parsing for Urdu [3]. They used MaltParser for training and tuning on the Urdu dependency treebank (UDT). Multi-level and multi-representational annotations like part-of-speech, chunking and dependency relation representation were used in developing this treebank [2]. There were 2853 sentences averaging length of 14.03 words in this treebank. The developed treebank had approximately 40012 tokens [2]. The selected corpus of UDT was constituent of all type of sentences, simple or complex. The corpus was manually annotated. It contained 35 POS tags, 9 chunk (phrase level) tags and small set of dependency relations [2].

UDT used only six dependency structures, which were the subject (subj), the object (obj), the secondary object (obj2), the adjectival modification (jmod), the adverbial modification (rmod) and the noun modification (nmod) [3].The authors converted the treebank data to CoNLL data format which was given to MaltParser as given input [3]. The selected CoNLL fields for the feature models contained were: ID (token counter, initialized by 1 for each new sentence), FORM (word form), CPOSTAG (coarse-grained part-of-speech tag), and POSTAG (Fine-grained POS tag), and the HEAD (head of the current token [3], which is a value of ID). The given input was in CoNLL format that is helpful for data-driven parsers. As a head of sentence Zero (0) is used and for phrase's head the value of ID is used and DEPREL (Dependency relation to the HEAD) [3]. The CoNLL format and annotation is demonstrated in Figure 4, by a typical sentence example [3].

(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."

Table 1. CONLL formate of the above sentence

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

The authors performed a series of experiments for evaluation of Urdu dependency parsing system. They performed all experiments using the Maltparser's default algorithm but each time selecting different feature models [3]. Their initial base line feature model used for Urdu Dependency Parsing was simple and it consists of word position, word, head and dependency relation. After that they enhanced the feature model by adding POS and chunk information [3]. The overall best achieved labeled accuracy (LA) for their experiments was 74.48% while for unlabeled attachment score (UAS) was about 90.14% that was achieved successively [3]. After that they manually did parsing to analyze and classify the different

types of errors produced by the parser. The obtained manual results were then compared with the treebank test data [3]. Another experiment was performed in 2012 by Ambati et al, in which they split UDT into training, testing and development sets. There were 2258 sentences in the training set and it was about 70% of total sentences in the corpus. Their test sets contained 484 that were about 15% of the total corpus. For sentence selection they used stratified partitioning technique. They tried to ensure that the three sets had approximately the same distribution of the parameters. They also used the "nivre-eager"and "nivre standard" parsing algorithm with default SVM setting parameters. They achieved 76.61 were76.61% LA, 88.45% UAS and 80.03% labeled accuracy scores using MaltParser. They desired to add chunk heads to check intra-chunk dependencies. Some other experiments had also been performed in many studies like [4], [5], [6], [8], [10], [11], [12], [13], [15], [16], [17], [20], [21] for various languages and results are reported in favor of the MaltParser.

5. PROBLEM STATEMENT

The treebanks are being built and checked against the different parsing mechanism specially the phrase structure and dependency structure. After this evaluation check their usefulness is normally reported. As for as, the phrase structure is concerned the URDU.KON-TB treebank has been evaluated and results are reported in [16]. However, this treebank has not been evaluated in the dependency parsing domain. The URDU.KON-TB treebank has the hyper dependency structure (HDS) encoded in it, which we will have to extract and parse accordingly. This evaluation will finally give us the answer that the URDU.KON-TB treebank is suitable for phrase or dependency parsing comparatively. To use the URDU.KON-TB treebank, we will have to convert its annotated data according to the format of the MaltParser's input e.g. CONLL format. As we are going to develop and experiment the computational resources in the domain of dependency parsing, so our research objectives/tasks will include the following

1. We will measure the compatibility of the URDU.KON-TB treebank w.r.t the format of the MaltParseras the treebank has the HDS annotation scheme encoded in it [16] and then we will claim its usability.

2. We will extract the data in CONLL format from the URDU.KON-TB treebank [1], so that it can be used to train the MaltParser.

3. We will perform experiments between the extracted data of the URDU.KON-TB treebank [1] and the MaltParser, and then finally report the suitability of the treebank in the domain of data-driven dependency parsing.

6. METHODOLOGY AND RESULTS

Computational model is an Urdu Dependency Parsing System [2]. We have used proposed Data-Driven Dependency Parsing computational model for Urdu language [3] on URDU.KON-TB [1].

6.1 Data Conversion process

In dependency treebank, the precedence order of annotation from lexical level to dependency level is POS > Syntactic > Dependency Relation which is a semantic relation between Head and Dependent as discussed in [3]. While the precedence order of the URDU.KON-TB treebank annotation from lexical level to functional level is the POS > Syntactic > 1st Division of the Functional tag > 2nd Division of the Functional tag [1]. 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). 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 [1]. Although URDU.KON-TB contains semantic information at each level proposed as future work (morpho-syntactic and syntactic-semantic) was proposed [2] but treebank does not contain dependency relation according to dependency grammar rules. So, we assumed the functional tagset as a Dependency Relation which are against the dependency. As Head information is also not available in the URDU.KON-TB treebank to run in MaltParser. According to treebank, we considered each word in the treebank is not head of any other word and nor dependent of any other word in the sentence. So, we assigned index value of token (ID) to the Head value in CONLL format. All these assumptions were made to make URDU.KON-TB data compatible and usable to run in MaltParser. But these assumptions did not fulfill the requirement of Dependency Treebank.

To convert the URDU.KON.TB tree to CoNLL based tree representation, we divide tree into multiple layers which represent columns in the CoNLL format. We defined following process or rules of conversion

- A layer is added at the bottom of the tree (below of edge leaves). This layer is named as ID in CoNLL. ID is an index of token which is an integer starting from 1 at each new sentence.
- Each word of the sentence is called FORM in CoNLL format.
- Semi-Semantic POS (SSP) tagged is listed under a layer called POSTAG, a column in CoNLL. It may contain POS with semantical and morphological subcategories but categories are optional.
- The next layer is based on Semi-Semantic Syntactic (SSS) annotation. This layer is called CPOSTAG. It is syntactic tag with morphosyntactic information and addition information is optional.
- The HEAD of each word is required in dependency parsing which is not available in URUDU.KON-TB. We considered each word in the treebank is not head of any other word and nor dependent of any other word in the sentence. So, we assigned index value of token (ID) to the Head value to make compatible w.r.t CoNLL format.
- Functional (F) tagset is called DEPREL. As there are two division level of functional tagset. We will keep in mind immediate functional tag to syntactic tag is consider as DEPREL otherwise mark as '_'. In case of '_' information is missing.

Here we will show conversion process to CONLL format by taking a sample sentence. Rules implementation and data extraction to save in CONLL to use in parsing.

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

Hamid killed the lion with a gun in the jungle.

Here is graphic representation of the given sentence which showing Semi-Semantic POS (SSP), Semi-Semantic Syntactic (SSS) and Functional (F) tagset [16].

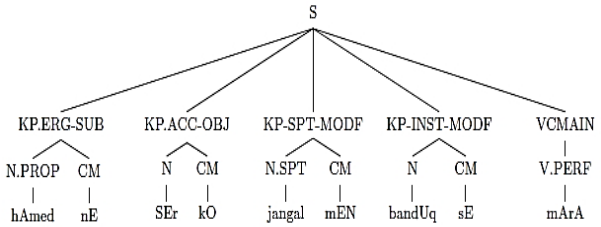

Fig 4: URDU.KON.TB representation

By applying above rules which were prose above, we will get the following representation of an URDU.KON.TB's sentence.

An example of a complete syntactic tag along with the functional labels according to the precedence order of the annotation for word 'hAmed' is N.PROP > KP.ERG > KP.ERG-SUB. Here, N.PROP is POS and a proper Noun, a syntactic tag KP with morphosyntactic information of ERGatove case (.ERG) is syntactic part, - SUB from the 2nd division of the functional tags is giving us information that KP is acting as subject of the sentence. Here 1st division of the functional tag is missing. N.PROP tag is listed under the POSTAG, KP.ERG is listed under the CPOSTAG and functional tagset is listed under DEPREL although it is against the dependency. From the above tree representation, we get following CoNLL format. This tree data saved in a .conll file to train and test the data in MaltParser dependency system.

This evaluated during the manual conversion process, the dependency relations and Head information is not available in the treebank. To make treebank compatible w.r.t to CONLL, we had made few assumptions to parse the data in MaltParser. So, there is issue of compatibility and data usability of URDU.KON-TB treebank in the dependency parsing domain.

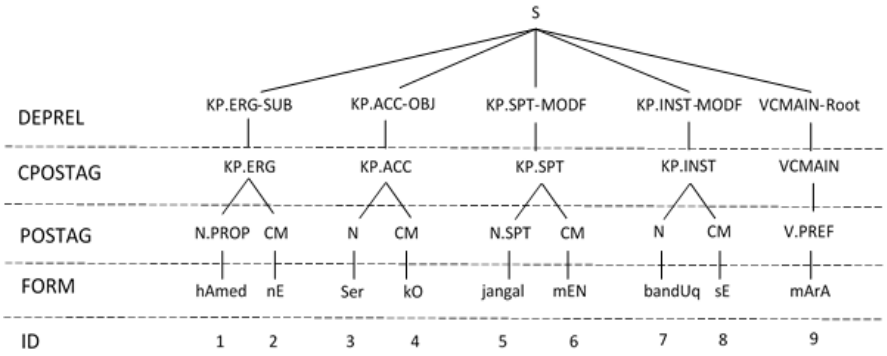

Fig 5: CoNLL format based treebank representation

Table 2. CoNLL file format of given sentence

ID	FORM	-	POSTAG	CPOSTAG	-	HEAD	DEPREL	-	-
1	حامد	-	N.PROP	KP.ERG	-	1	KP.ERG.SUB	-	-
2	نے	-	CM	KP.ERG	-	2	KP.ERG.SUB	-	-
3	شیر	-	N	KP.ACC	-	3	KP.ACC.OBJ	-	-
4	کو	-	CM	KP.ACC	-	4	KP.ACC.OBJ	-	-
5	جنگل	-	N.SPT	KP.SPT	-	5	KP.SPT.MODF	-	-
6	میں	-	CM	KP.SPT	-	6	KP.SPT.MODF	-	-
7	بندوق	-	N	KP.INST	-	7	KP.INST.MODF	-	-
8	سے	-	CM	KP.INST	-	8	KP.INST.MODF	-	-
9	مارا	-	V.PERF	VCMAIN	-	9	VCMAIN.Root	-	-

6.2 Architecture

The architecture of our Dependency Parsing System (DPS) is presented in figure 6. Input of the system is URDU.KON-TB which was based on the bracketing notation. As we are using MaltParser, so we need to convert this tree to CoNLL format first then we can use converted data into the system. A complete conversion process by defining conversion rules is proposed here.

The conversion rules applied manually and data is prepared in CoNLL format, then this data is passed to the Dependency Parsing system which uses MaltParser. MaltParser is trained on this dependency tree which annotated by SSP, SSS and F tagset. Then the sentence is parsed to get the dependency relation (DEPREL).

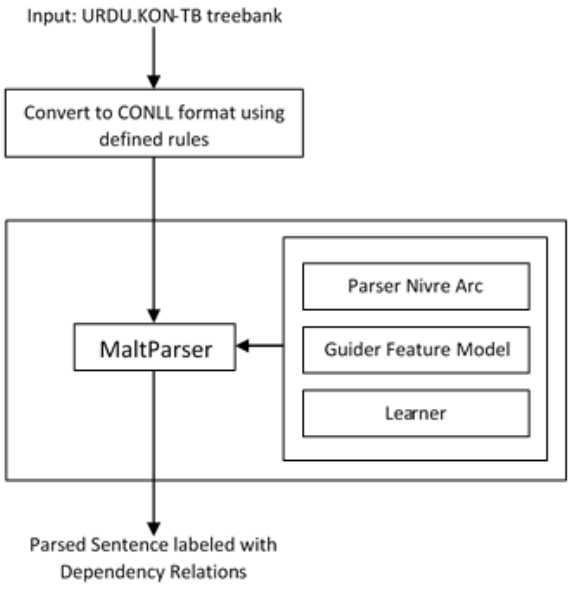

Fig 6: Dependency Parsing System

6.3 Experiments

The series of experiments performed to evaluate treebank in the Urdu dependency parsing system. The treebank URDU.KON.TB data is converted to CoNLL format, and then converted data after splitting it to trained data and tested data is given to MaltParser. Nivre arc-eager algorithm is used for parsing in training and testing but with different feature models as proposed by Ali [2]. Six different feature models are used [2]. In initial, a simple feature model containing ID, FORM, assumed head of current token and DEPREL (assumed Functional (F) tagset) of current token is used. The next experiment is performed by extending the feature model and semi-semantic POS (SSP) added. In third experiment, the feature model extended by adding semi-semantic Syntactic (SSS). In the last experiment, both included. So, the total numbers of four experiments are performed [2] which are listed in the table 3.

Table 3. Experiments

No.	Parsing Algorithm	Feature Model
1.	Nivre-arc-eager	ID, FORM, HEAD, DEPREL (F)
2.	Nivre-arc-eager	ID, FORM, POSTAG (SSP), HEAD, DEPREL (F)
3.	Nivre-arc-eager	ID, FORM, CPOSTAG (SSS), HEAD, DEPREL (F)
4.	Nivre-arc-eager	ID, FORM, POSTAG (SSP), CPOSTAG (SSS), HEAD, DEPREL (F)

To check the data of URDU.KON-TB is usable for developing a new dependency treebank, we make an assumption based enhancement. We assume, every word in a sentence is Head of the sentence although it is not following dependency grammar rules. But the aim is to check the usability of data. So, the value of Head will become zero. Same four experiments which are listed in table (3) are performed with new assumed Head value.

6.4 Result

MaltParser was trained and tested on URDU.KON-TB Treebank converted data. As the given URDU.KON-TB contains 25 sentences with average length of 15 words. We prepared the training data by applying rules which were proposed to convert. Then the system was trained with 80% data and tested on 20% data. The series of four experiments were performed which are based on different feature models. The accuracy and correctness of DEPREL tag is calculated by comparing MaltParser parse output with manually tagged test data. The accuracy percentage of each experiment is calculated using this formula:

$$Accuracy\ (\%) = \left(\frac{Total\ No.of\ correct\ marked\ DEPREL}{Total\ No.of\ tokens}\right) X\ 100$$

The accuracy of zero percent is noted in first four experiments. As the given treebank was based on phrase structure and do not contained dependency head and dependent information. So, the MaltParser is unable to parse without dependency relation with head and dependent information. On the bases of results, we can say that URDU.KON-TB treebank is not suitable for the dependency parsing domain. The results also show increasing the features in the model is not helpful to increase the accuracy until the head and dependent information is not available in the treebank. To support our finding and argument, we assumed that every word in a sentence is Head of the sentence and four experiments performed to check URDU.KON-TB is not suitable for dependency due to information missing according to dependency grammar. The result of experiments listed in table 4, show MaltParser parsed and some reasonable accuracy also noted. So, it is evaluated URDU.KON-TB is not suitable for the dependency parsing domain but the data of this treebank (SSP and SSS) is usable to develop a new treebank which will be a dependency treebank.

Table 4. Results

Experiments with Feature Model	Accuracy (%)	
	Default Experiment	Enhance Experiment
ID, FORM, HEAD, DEPREL (F)	0	22
ID, FORM, POSTAG (SSP), HEAD, DEPREL (F)	0	22
ID, FORM, CPOSTAG (SSS), HEAD, DEPREL (F)	0	49
ID, FORM, POSTAG (SSP), CPOSTAG (SSS), HEAD, DEPREL (F)	0	49

6.5 Discussion

The URDU.KON-TB was annotated from lexical level to functional level (POS, Syntactic and Functional) which is suitable for the phrase structure parsing because all these tags related to constituents. The semantic information is available on each level as future work (morpho-syntactic and syntactic-semantic) was proposed [2] but treebank did not include dependency relation which are required for the dependency parsing. Head and dependent are not marked in the treebank. The immediate relation of head and dependent is also not available. The functional tags are just constituents' tags. The converted data based on assumptions was trained and parsed

using MaltParser. The result clearly shows. URDU.KON-TB treebank is not suitable for dependency parsing domain. Then by adding assumption based Head information and the results of new experiment validate our argument. From new experiments, we also concluded, some data of the treebank (SSP and SSP) is usable in making dependency treebank. It is also evaluated during conversion process due to missing information; URDU.KON-TB treebank data is also not fully compatible to CONLL format. Head information added on assumption to make it compatible. In same way, as dependency relations are not available to parse in MaltParser, it was assumed the Functional tags are DEPREL which is against the dependency grammar and treebank. There are also issue of annotation level; as per dependency, there should be immediate tags relation. But it is observed, there is no immediate a functional tag available above the syntactic tag in URDU.KON-TB.

7. CONCLUSION

In this paper, our task was to evaluate URDU.KON-TB in the dependency parsing domain. As for as, the phrase structure is concerned the URDU.KON-TB treebank has been evaluated and results are reported in [16]. To use the URDU.KON-TB treebank, we have converted the annotated data according to the format of the MaltParser's input e.g. CONLL format by proposing rules. During the conversion process, we also checked the treebank data compatibility w.r.t to CoNLL and usability of data. A few assumptions were taken to make the data compatible in MaltParser although these assumptions were against the dependency grammar rule. Nivre arc-agear algorithm is used in parsing system to train and test the data with six different feature models and four experiments [2]. To validate our findings, we conduction four more experiment using assumption based Head information although information was not following dependency grammar. The results show some accuracy due to Head information and support our argument, the URDU.KON-TB does not have information which are required for the dependency parsing domain. It is not suitable for the dependency parsing domain. In future work, we can use some data of URDU.KON-TB to develop a new dependency treebank. The SSP and SSS information can be used but we need all other effort which are required for dependency treebank, the head dependent relationship. Then the functional tagset can be marked by following the dependency grammar rules which are more than constituents' tags. We can also enhance it by adding boundary of phrases as proposed [2]. In this way, we can conduct more experiment.

8. ACKNOWLEDGMENTS

We are thankful to **Sir Wajid Ali** for working with us.

9. REFERENCES

[1] Abbas, Q. (2014). Building Computational Resources: The URDU. KON-TB Treebank and the Urdu Parser (Doctoral dissertation).

[2] Ali, W., &Hussain, S. (2010). Urdu dependency parser: a data-driven approach. In Proceedings of Conference on Language and Technology (CLT10), SNLP, Lahore, Pakistan.

[3] Ali, W, (2010). Data-Driven Dependency Parsing for Urdu, MS (MPhil), Computer Sciences thesis, Department of Computer Sciences, National University of Computer and Emerging (NUCES), Lahore, Pakistan.

[4] Bhat, R. A., Jain, S., & Sharma, D. M. (2012). Experiments on dependency parsing of Urdu. *Proceedings of TLT11*, 31-36.

[5] Bhat, R. A., & Sharma, D. M. (2012, July). A dependency treebank of Urdu and its evaluation. In *Proceedings of the Sixth Linguistic Annotation Workshop* (pp. 157-165). Association for Computational Linguistics.

[6] Abbas, Q. (2014). Semi-semantic part of speech annotation and evaluation.*LAW VIII*, *75*.

[7] Nivre, J., Hall, J., & Nilsson, J. (2006, May). Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC* (Vol. 6, pp. 2216-2219).

[8] Bharati, A., Husain, S., Ambati, B., Jain, S., Sharma, D., &Sangal, R. (2008). Two semantic features make all the difference in parsing accuracy.*Proc. of ICON*, *8*.

[9] Ballesteros, M., &Nivre, J. (2012, May). MaltOptimizer: A System for MaltParser Optimization. In *LREC* (pp. 2757-2763).

[10] Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., ... &Marsi, E. (2007). MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, *13*(02), 95-135.

[11] Bohnet, B., &Nivre, J. (2012, July). A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (pp. 1455-1465). Association for Computational Linguistics.

[12] Spreyer, K., & Kuhn, J. (2009, June). Data-driven dependency parsing of new languages using incomplete and noisy training data. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning* (pp. 12-20). Association for Computational Linguistics.

[13] Ambati, B. R., Husain, S., Nivre, J., &Sangal, R. (2010, June). On the role of morphosyntactic features in Hindi dependency parsing. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages* (pp. 94-102). Association for Computational Linguistics.

[14] Nilsson, J. (2009). Transformation and Combination in Data-Driven Dependency Parcing.

[15] Nivre, J. (2008). Sorting out dependency parsing. In *Advances in Natural Language Processing* (pp. 16-27). Springer Berlin Heidelberg.

[16] Abbas, Q. 2015, Morphologically rich Urdu grammar parsing using Earley algorithm, Natural Language Engineering (NLE), Vol.21(2), PP.1-36, ISSN: 1351-3249, DOI: 10.1017/S1351324915000133, Cambridge University Press, UK

[17] N. Chomsky. Three Models For The Description Of Language. Information Theory, IRE Transactions on, 2(3):113–124, 1956.

[18] PUNEETH, K. (2016). Dependency Parsing and Empty Category Detection in Hindi Language (Doctoral dissertation, International Institute of Information Technology Hyderabad).

[19] GADE, R. P. (2014). Dependency parsing approaches for Indian Languages: Hindi and Sanskrit (Doctoral dissertation, International Institute of Information Technology Hyderabad).

[20] J. Nivre, Inductive Dependency Parsing, Springer, 2006.

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