Towards building a Kashmiri Treebank: Setting up the Annotation Pipeline

Riyaz Ahmad Bhat♣, Shahid Mushtaq Bhat♠, Dipti Misra Sharma♣

LTRC, IIIT-Hyderabad, India♣

LDC-IL, CIIL Mysore♠

riyaz.bhat@research.iiit.ac.in, ldc-shahid@ciil.stpmy.soft.net, dipti@iiit.ac.in

Abstract

Kashmiri is a resource poor language with very less computational and language resources available for its text processing. As the main contribution of this paper, we present an initial version of the Kashmiri Dependency Treebank. The treebank consists of 1,000 sentences (17,462 tokens), annotated with part-of-speech (POS), chunk and dependency information. The treebank has been manually annotated using the Pāṇinian Computational Grammar (PCG) formalism (Begum et al., 2008; Bharati et al., 2009). This version of Kashmiri treebank is an extension of its earlier version of 500 sentences (Bhat, 2012), a pilot experiment aimed at defining the annotation guidelines on a small subset of Kashmiri corpora. In this paper, we have refined the guidelines with some significant changes and have carried out inter-annotator agreement studies to ascertain its quality. We also present a dependency parsing pipeline, consisting of a tokenizer, a stemmer, a POS tagger, a chunker and an inter-chunk dependency parser. It, therefore, constitutes the first freely available, open source dependency parser of Kashmiri, setting the initial baseline for Kashmiri dependency parsing.

Keywords: Treebanks, Computational Resources, Dependency Parsing

1. Introduction

The need for manually annotated linguistic resources is widely acknowledged in the field of computational linguistics. Due to their importance for basic as well as advanced NLP applications, the last decade has seen nearly an exponential increase in the creation of these linguistic resources in a wide range of languages. Syntactic treebank is one such important resource which has seen a rigorous development among the world languages due to its widespread usage. A syntactic treebank is, by definition, a set of syntactic trees capturing the syntactic or semantic structure of sentences. Creation of these treebanks has interested both linguists and computational linguists. For the former, they provide insights about the linguistic theory they have been built upon, and the later use them for the development of data driven parsers. Usually, treebanks are multi-layered. At every layer a separate but related set of linguistic information is marked. Annotation at the lower layer facilitates the annotation at the higher layer. In this way, treebank development follows a pipeline approach, with different levels of linguistic information annotated one after the other. In this paper, we setup a treebanking pipeline for Kashmiri following a treebanking pipeline for Indian languages. We also present the basic computational tools used in the pipeline and discuss some of the theoretical issues concerning the Kashmiri treebanking.

The paper is organized as follows. Section 2 provides a quick overview of Kashmiri - its genealogy and special grammatical features. Section 3 describes the Indian Language treebanking pipeline. Section 4 discusses annotation of Kashmiri corpora based on the IL annotation pipeline. It also discusses the experimentation for the creation of tools using the treebank data. Finally, Section 5 concludes the paper with some future directions.

2. About Kashmiri

Kashmiri language belongs to the Dardic sub-group of the Indo-Aryan family. It is spoken primarily in the Kashmir Valley, in Jammu and Kashmir. According to the census of 2001, it has approximately 5,632,698 speakers throughout India and Pakistan. It is one of the 22 scheduled languages of India[1].

Kashmiri is a V2 language like German in which tensed clauses are subjected to verb second constraint. The finite verbal element in these clauses always occurs in the second position, i.e., the position immediately following the first phrasal constituent (Hook and Manaster-Ramer, 1985; Bhatt, 1995; Bhatt, 1999). In the cases where there is an auxiliary verb carrying tense information, it occupies the second position in the clause but the main verb occupies the final position. Consider examples 1 and 2 for an illustration of $v2$ phenomenon in Kashmiri. In example 1, auxiliary chu 'is' occupies clause second position while the verb $diwan$ 'give' occurs at the end. Tensed verb $dits$ 'give' follows the first phrasal constituent in example 2.

(1) رَام **چھ** شَامَس کِتَاب **دِوَان**

ram	***chu***	*shamas*	*kitab*	***divan***
Ram	be+prs	Sham+dat	book	give+prog

'Ram is giving a book to Sham.'

(2) رَامن **دِژ** شَامَس کِتَاب

ram	***dits***	*shamas*	*kitab*
Ram+erg.	give+pst.	Sham+dat	book

'Ram gave Sham a book.'

[1] http://en.wikipedia.org/wiki/Kashmiri_language

Kashmiri is inflectionally rich language. The morpho-syntactic information, like 'person', 'number', 'gender' and 'case', is realized through a portmanteau morph. Among the major word classes, nouns decline for number, gender and case, and verbs conjugate for tense, aspect and modality (TAM). Apart from TAM, verbs carry agreement features of one of their arguments and can also undergo different morphological processes such as passivization and causativisation. Similar to English, tense information either occurs on the main verb or can stand as a free word in the form of tense auxiliaries.

3. Indian Language Treebanking Pipeline

The dependency treebanks for Indian Languages based on CPG formalism are developed following a generic annotation pipeline. The process of treebank development under the pipeline consists of a series of steps namely (i) Tokenization, (ii) Morph-Analysis, (iii) POS-tagging, (iv) Chunking, and (v) Dependency annotation. Annotation process begins with the tokenization of raw text. The tokens so obtained during the process are annotated with morphological and POS tag information in the next steps. After morph-analysis and POS-tagging locally dependent contiguous tokens are grouped together into chunks. The text processing mentioned in these steps has been automated with the help of a battery of NLP tools, namely tokenizer, morph analyzer, POS-tagger and chunker, which have been developed in-house and have pretty good accuracy. The output of each tool is manually corrected and validated by human annotators. The final step in the pipeline is the manual dependency annotation. Only the inter-chunk dependencies are marked leaving the intra-chunk dependencies unspecified. The strategy has been adopted to reduce the time consuming manual labor which is hallmark of syntactic annotations. The intra-chunk dependencies are made explicit automatically at a later stage of the treebank development. This strategy is motivated by the fact that intra-chunk relations are highly predictable and can be automated if the chunk boundaries and the chunk heads are specified.

As aforementioned, we follow the annotation pipeline for Kashmiri treebanking. We will discuss the development of various tools used in the pipeline so that the semi-automated nature of the annotation process can be justified. The complete toolkit can be downloaded from here[2].

3.1. Tokenisation

Tokenisation is a relatively easy task for the languages written in Roman script. However, the task becomes quite complex for languages written in other scripts, particularly, for the languages using persio-arabic script. Persio-arabic script poses two problems to tokenisation; namely, **space omission** and **space insertion**. A space character has hardly any significance in visual word identification as a word boundary marker, thus, it can be omitted altogether. It is needed to generate the correct typography of a word (Durrani and Hussain, 2010) which has considerable role in readability of the text. However, due to the impact of technology which, by and large, is itself under the impact of English, the space character has become more or less a standard word boundary marker. Therefore, space character has now acquired two functions in languages written in Persio-Arabic script: to separate words and to generate correct typography. In the current work, following Urdu treebanking pipeline, text is tokenized using the space character as word boundary marker. Human annotators, while validating the output of the morphological analyzer, correct the wrong segmentation and join the word segments using underscore "_" . At a later stage, the underscores "_" are replaced with zero width non-joiner character "ZWNJ"[3], which converts the text into its natural form (by removing the extra "_" character) and addresses the **space insertion** problem. In the future, we will attempt to automate the identification of words which are split in multiple tokens for correct typography, so that the text tokenisation can be made more reliable without human intervention.

3.2. Morph Segmentation

Kashmiri is inflectionally rich language with nouns and verbs inflecting for different set of grammatical information. Nominals inflect for number, gender and case which are realized through a single portmanteau morph, e.g. in the noun, *insaan-an* (human-Erg.SG.M) '*-an*' is an ergative marker which also carries singularity and masculinity information with it, hence, a portmanteau morph. Further, nominal modifiers like demonstratives, quantifiers and adjectives agree with their head noun for the number, gender and case, e.g. in the NP, *yam-is ak-is bad-is ladak-as* (this-DAT.SG.M one-DAT.SG.M big-DAT.SG.M boy-DAT.SG.M) all the dependent words (modifiers) agree with the head $ladake$ 'boy' in terms of number, gender and case information which is condensed on a single dative marker (-is/-as). Besides the inflectional information, Kashmiri nouns and verbs also have a derivational morphology (for more details see (Koul and Wali, 2006)). Therefore, automatic segmentation of such markers from their roots can be done easily. Similarly, verbs carry tense, aspect and mood (TAM) information and show agreement properties. Kashmiri verbs can also be inflected for emphatic, honorific, negative and interrogative markers. In addition to these pragmatic markers, the verbs exhibit an interesting phenomenon of pronominal cliticization. The clitic may be only subject enclitic, only object enclitic or both. these clitics show PNG agreement with the corresponding argument, e.g. *shong-us* (slept-1PC.SG.M = I-M slept). Furthermore, Kashmiri has morphological causatives and passives, i.e. causatives and passive forms are derived by clean morphological process. Suffixation of '*-inaav*' to the root form produces a causative form and suffixation of '*-ni*' produces passive form. All these markers can be easily segmented with hardly any need to compensate morphophonemic changes. In the Indian language treebanking pipeline, a paradigm based morph analyzer is used for the morph analysis. Such a tool, however, has not yet been built for Kashmiri. As an

[2]http://researchweb.iiit.ac.in/ riyaz.bhat/Resources/parser.tar.gz

[3]http://en.wikipedia.org/wiki/Zero-width_non-joiner

alternative, we use an unsupervised morphological parsing algorithm (Dasgupta and Ng, 2006) for the morph analysis of Kashmiri text. The accuracy of the algorithm run of 13,585 unique words is 72.73%. In the future, we plan to build a high coverage paradigm based morph analyzer for Kashmiri.

3.3. POS-Tagging

We have used the current version of Indian Language Machine Translation (ILMT) pos tag set for POS tagging the Kashmiri corpus without any additional changes (Bharati et al., 2006). We have manually tagged around 61,741 words (3,409 sentences). The tagged corpus is used for building a statistical POS tagger for Kashmiri. The data set is split into training and testing by a ratio of 80:20. We used CRF++ (Lafferty et al., 2001) [4] package to build the POS tagger. In the baseline, we used a context window of 4 words; for every word capturing its context till 2 preceding and following words. To address the data sparsity and OOV problem caused by the rich morphological nature of Kashmiri, we used the automatic morph analysis. In the first experiment, we used the morph segmenter, discussed in Morph Segmentation above, to generate the stem and affixes of a word. We achieved an accuracy of 79.16%, an increase of 2.52% from the baseline. In the second experiment, we used the first and last 4 characters of a word. This increased the accuracy by 6.81% from the baseline.

	Baseline	Morph Segmentation	Morph Segmentation
Accuracy (%)	76.64	79.16	83.45

Table 1: POS-Tagging Accuracy

3.4. Chunking

We have manually chunked the already pos tagged corpus of 3,409 sentences (41,678 chunks). We are using the ILMT guidelines for chunking (Bharati et al., 2006), however, with significant changes addressing the verb-second (V2) phenomenon in Kashmiri. Unlike other Indian languages, auxiliaries in Kashmiri can stand away from the main verb in the second position of a clause as a separate constituent. The notion of a verb group, where verb and its auxiliaries act as a highly cohesive unit, is weaker in Kashmiri. We have introduced few additional chunk tags for verb and auxiliary projections not used in ILMT guidelines treating every verb and auxiliary as a separate unit rather than a part of a bigger verb group. The additional chunk tags are presented in Table 2.

S.No.	Tag	Description
1	AUXP	All Auxilaries
2	VCM	Main Verb with separate tense auxilary
3	VCF	Tense verb
4	VCNN	Gerund
5	VCNF	Non-finte Verb
6	VCINF	Infinitive

Table 2: New Chunk Tags

We used the manually chunked data for building an automatic chunker. We used the CRF++ for this purpose. The data is split by 80-20 for training and testing the chunker. We set the baseline with a simple context window of 2 preceding and following words. The chunking accuracies are improved with the introduction of POS tag information as shown in Table 3.

	Baseline	Gold POS-Tag	Automatic POS-Tag
Accuracy (%)	75.35	92.77	81.34

Table 3: Chunking Accuracy

3.5. Dependency Annotation

Among the 3,409 sentences manually POS tagged and chunked, we annotated 1,000 sentences (11,848 nodes/tokens) with the dependency structures. We are using the dependency annotation guidelines proposed in (Begum et al., 2008; Bharati et al., 2009). The guidelines are based on the Computational Pāṇinian Grammar formalism inspired by an ancient Indian grammarian named Pāṇini (Bharati et al., 1995). Figure (1) shows annotation of an example sentence based on the guidelines. The labels starting with 'k' are Pāṇini's Karaka relations which are central to CPG formalism. A Karaka relation is a grammatical relation that holds between a verb and its arguments or some adjuncts.

We also carried an annotator agreement study on a set of 100 sentences. The data set was separately annotated by two expert linguists. The agreement statistics shown in Table 4 suggests a good understanding of annotators of the annotation guidelines and the morpho-syntax involved in the given set of corpus. The major disagreement is observed for the major dependency labels namely k1 'agent/subject', k2 'patient/theme/object' and k1s 'noun complement' which indicates that the arguments which bear k1, k2 and k1s relations with a verb are most confusing grammatical relations in the treebank. It seems morpho-syntactic cues serve as poor guide in certain contexts. Some of the reasons for the disagreement are the size of a sentence (lower agreement for sentences with >50 words), higher degrees of argument scrambling, ambiguity in morpho-syntactic cues (case suffixes) and the lack thereof etc.

(3) بچھ مگتہ اَنُن چھُ سٹھَاہ پرون رسم ییمیٖک حوَالہ
اَسہ رَامَاینہء مز تہء مہَابہَارتہء مز تہِ مَیلَان چھُ

bachi-NOM manghtI an-un chu seThaa prA:n'
child adopt bring is very old
resIm yem'-uk hawaalI asi-DAT
custom whose-GEN.SG.M reference we
raamaayin-I manz-I tI mahabart-I
Ramayana-ABL in and Mahabharata-ABL
manz-I ti me'l-aan chu
in also get be-PRS.SG.M
'Child adoption is a very old custom reference of

[4] http://crfpp.googlecode.com/svn/trunk/doc/index.html?source=navbar

which we even get in the Ramayana and Mahabharata.'

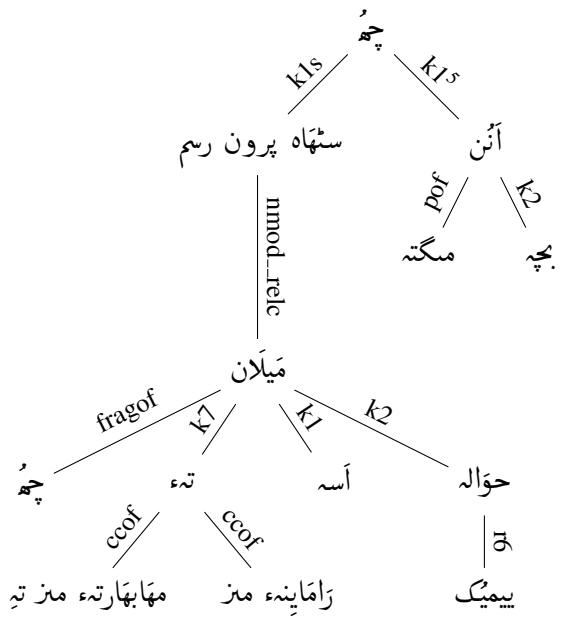

Figure 1: Dependency Tree of an Example Sentence

No. of Annotations	Agreement	P(a)	P(e)	κ
1132	880	0.777	0.089	0.756

Table 4: Kappa Statistics

The annotated corpora is used to build a first dependency parser of Kashmiri which we plan to use in future to bootstrap the treebank. The data set is split into training, testing and development sets by the ratio of 80-10-10 in the conll formatt. We used the MaltParser (Nivre et al., 2007) for parsing experiments. MaltParser uses a transition-based approach to dependency parsing. In order to select the best algorithm and tune the parameters of MaltParser, we used MaltOptimizer (Ballesteros and Nivre, 2012) on the training data. As suggested by MaltOptimizer, we experimented with Covington parsing algorithm with non-projective settings. Current version of Kashmiri treebank has 0.02 non-projective edges, which is way lower compared to other Indian languages (Bhat and Sharma, 2012). We experimented with different feature sets both gold as well as automated. The results are reported in Table 5.

Baseline for parsing is set using just the raw tokens. Auto Lemma and Auto POS features are extracted using the POS-tagger and Morph segmenter that we presented in this work. Since Kashmiri has very rich morphology, we either need a good lemmatizer or enough data to address the problem of data sparsity. Even though results are not that promising, we have set the baseline for further research on Kashmiri dependency parsing.

Features	**LAS (%)**	**UAS (%)**	**LAcc (%)**
Baseline	25.71	42.91	33.30
Auto Lemma	25.71	46.05	32.89
Gold POS	42.71	66.50	48.28
Auto POS	34.51	53.64	42.31
Gold Chunk	45.14	67.00	47.77
Gold POS, Gold Chunk	**46.36**	**70.55**	**49.49**
Auto Lemma, Gold POS, Gold Chunk	40.89	62.65	46.66
Auto Lemma, Auto POS	33.00	53.04	39.68

Table 5: Effect of different features on the Malt parser.

4. Conclusion

As a contribution to Kashmiri language, we presented two important computational resources for its text processing. These are:

1. A manually annotated Kashmiri dependency treebank, in which 3,409 sentences are POS-tagged and chunked while 1,000 sentences are annotated with dependency structures. We made few changes to the guidelines (Bharati et al., 2006) to address and accommodate the *v2* phenomenon in Kashmiri. To assure the quality of the treebank, we carried out an inter-annotator study. A kappa value κ=0.76 shows sufficient agreement between the 2 expert annotators and thus assures the quality of the Kashmiri treebank.

2. A dependency parsing pipeline, which includes, a tokenizer, a morph segmenter, a pos tagger, a chunker and an inter-chunk dependency parser. All tools perform well (>80) except the dependency parser, which suffers due to the lack of enough data used for its training.

For future work, we plan to annotate all the 3,409 POS-tagged and chunked sentences with dependency structures. Since the dependency structures in the current version of the treebank are annotated between chunk heads, we also plan to express the dependencies between tokens in a chunk.

5. References

Miguel Ballesteros and Joakim Nivre. 2012. Maltoptimizer: an optimization tool for maltparser. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 58–62. Association for Computational Linguistics.

R. Begum, S. Husain, A. Dhwaj, D.M. Sharma, L. Bai, and R. Sangal. 2008. Dependency annotation scheme for indian languages. In *Proceedings of IJCNLP*. Citeseer.

[5] $k1$ 'Agent', $k2$ 'Patient', $k1s$ 'Noun Complement', $r6$ 'genitive', $ccof$ 'co-ordination', pof, 'Part of a Complex Predicate', $nmod\_relc$ 'Relative Clause'

A. Bharati, V. Chaitanya, R. Sangal, and KV Ramakrishnamacharyulu. 1995. *Natural Language Processing: A Paninian Perspective*. Prentice-Hall of India.

Akshar Bharati, Rajeev Sangal, Dipti Misra Sharma, and Lakshmi Bai. 2006. Anncorra: Annotating corpora guidelines for pos and chunk annotation for indian languages. *LTRC-TR31*.

A. Bharati, D.M. Sharma, S. Husain, L. Bai, R. Begum, and R. Sangal. 2009. Anncorra: Treebanks for indian languages guidelines for annotating hindi treebank (version–2.0).

Riyaz Ahmad Bhat and Dipti Misra Sharma. 2012. Non-projective structures in indian language treebanks.

Shahid Mushtaq Bhat. 2012. Introducing kashmiri dependency treebank. In *24th International Conference on Computational Linguistics*, page 53.

Rajesh Bhatt. 1995. Verb movement in kashmiri. *U. Penn Working Papers in Linguistics*, 2.

Rakesh Mohan Bhatt. 1999. *Verb movement and the syntax of Kashmiri*. Kluwer Academic Publishers.

Sajib Dasgupta and Vincent Ng. 2006. Unsupervised morphological parsing of bengali. *Language Resources and Evaluation*, 40(3-4):311–330.

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

Peter Edwin Hook and Alexis Manaster-Ramer. 1985. The position of the finite verb in germanic and kashmiri. towards a typology of v2 languages. In *Germanic Linguistics: Papers from a Symposium at the University of Chicago*, pages 46–58.

O.N. Koul and K. Wali. 2006. *Modern Kashmiri Grammar*. Dunwoody Press.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.