Enhancing Semantic Role Labeling in Hindi and Urdu

Aishwary Gupta, Manish Shrivastava
IIIT Hyderabad, India
aishwary.gupta@research.iiit.ac.in, m.shrivastava@iiit.ac.in

Abstract

This paper presents a supervised semantic role labeler for Hindi which can be extended to Urdu as well. We propose a set of new features enriching the existing baseline system for these languages. We break the system into two subsequent tasks - Argument Identification and Argument Classification respectively. Our experiments show a reasonable improvement with respect to the current baseline for Hindi, mainly for the classification step. We also report significant improvements for Argument Identification task in Urdu. Finally, we create a new baseline for the Hindi using 5-fold cross-validation and we capture results excluding the null class and including the null class exclusively. We also extend the same work on Urdu and report the results.

Keywords: Semantic Role Labeling, PropBank, TreeBank, CBOW, Word2Vec

1. Introduction

In the last decade, there has been a lot of interest and a great amount of contribution towards semantic analysis of languages. There has been a significant amount of work done for major languages like English which included efforts in making semantically annotated data like PropBank. But only a little effort has been shown in Indian languages such as Hindi and Urdu. The Hindi PropBank and the Urdu PropBank were proposed just a few years back following which the first system, a semantic role labeler(SRL) for these languages was built. We saw a need for improvement in this domain and thus we introduce a new system with an additional set of features which makes a significant improvement in the classification of semantic roles for Hindi and extend the same work for Urdu. The major objective of SRL is to provide all sorts of information from a sentence in the form - who does what, where, to whom, where, when etc. In the PropBank, each sentence can be thought of as an event(s) which has participants - analogous to predicate having arguments. This labeling is done at phrase(chunk) level. The verb is the predicate and phrases/chunks related to it are its arguments labeled in categories such as Doer, Receiver, location, temporal etc. A SRL system has to therefore, label the arguments for each predicate of a sentence automatically.

Most of the previous works like (Pradhan et al., 2005), (Punyakanok et al., 2004), (Koomen et al., 2005) and (Anwar and Sharma, 2016) use a 2 step approach, i.e., first a chunk is identified whether it is an argument for a given predicate in the sentence or not. If yes, then it is classified at second step into the role labels. We use the same approach for reasons given in Section 3.1.

The applications of SRL can be seen in various research areas in Natural Language Processing(NLP). It can be attributed to the fact that semantic role labeling provides the meaning of a sentence at an abstract level. It can be seen in fields like information extraction (Christensen et al., 2010), question answering (Pizzato and Mollá, 2008) and machine translation (Liu and Gildea, 2010). Our paper is organized as follows:

Section 2 gives a brief description about the Hindi and the Urdu PropBank and how these were annotated above their respective TreeBanks. We also talk about the language resource we used for our task. Section 3 gives an idea about the related work done for this task. Section 4 shows our detailed approach and the system architecture. It also talks about the classifier we have used which helps us cut down the argument-identification task. In Section 5, We talk about the current best system as the baseline and then talk about the new features we have proposed. In section 6, we show how we conducted our experiments and the results for both languages. This also includes the comparison of our system with the existing system.

2. The Hindi Propbank and The Urdu Propbank

The Hindi Treebank and The Urdu Treebank were added with a layer of semantic annotation to give rise to The Hindi Propbank (Vaidya et al., 2011) and the Urdu PropBank (Anwar et al., 2016) respectively. These are part of the Hindi-Urdu PropBank Project which is a multi-dimensional and multi-layered resource creation effort for the Hindi and the Urdu language (Bhatt et al., 2009). Unlike PropBanks in most of the other languages, these PropBanks are annotated on top of the corresponding TreeBanks which have a dependency structure. The Treebank already having the dependency annotation, now including lexical semantic information at chunk level forms the PropBank corpus. PropBanks of both the languages include dependency relations at the chunking level which help construct the sentence dependency tree, morphological information for each word, part-of-speech/syntactic category at chunk as well as token level. The PropBanks as well, similar to the TreeBanks, are represented in the Shakti Standard Format (Bharati et al., 2007). The sentences are distributed in various documents. Each document has 15-20 sentences where each sentence is broken down into chunks and each chunk is broken down at token level.

Propbank labels‘ (or semantic role labels) annotation was made easy by dependency relations - also called as karaka relations (Vaidya et al., 2011) (described later) because there is a close syntactic-semantic relation in them. In the PropBank, semantic roles are defined for each verb which means that a fixed set of roles are specified for each verb

and a distinct label is assigned to each role. These are labeled in different ways in various PropBank annotations. For Hindi, the core arguments are the numbered arguments which are labeled as ARG# where #-{0,1,2,3}. For example, the verb bawA(to tell), has four such numbered arguments: ARG0: person telling, ARG1: thing told, ARG2: hearer there is no ARG3 for this verb. An important point to be noted here - an argument marked with the same number, say. ARG0, may not share any semantic similarities for different verbs. Further, each verb can have a different set of these arguments depending on the use of the verb in a sentence. This is handled by providing different frameset/sense to each verb which means that the annotation also has the information of which way the verb is being used in a sentence. Example- the same verb bawA - has another meaning which is - to mention/describe and hence has a slight difference in its set of arguments, namely, ARG0: the one mentioning or describing A as B, ARG1: the thing A that is described as B, ARG2-ATR: the description B that is used to describe A. The Hindi PropBank has distributed ARG2 into 4 more labels namely - ARG2-ATR(attribute), ARG2-LOC(location), ARG2-GOL(goal), ARG2-SOU(source). There are also certain other modifier labels denoted as ARGM* which are not specific to any verb and can be shared by any verb. The Hindi PropBank has 24 distinct labels and the Urdu PropBank has 20 distinct labels with number of modifiers being 4 less than those in Hindi.

2.1. Dataset

As reported earlier (Anwar and Sharma, 2016), for Hindi PropBank they took around 100,000 tokens as training data and 20,000 as test data and for Urdu they took 130,000 tokens as training data and 30,000 as test data. We have used exactly same data for Phase 1(Section 6) of our experiments.

3. Related Work

According to the best of our knowledge, only work done on automatic semantic role labeling for Indian Languages PropBank, i.e., Hindi PropBank and Urdu PropBank was seen last year (Anwar and Sharma, 2016). Other than this, on English PropBank, plenty of work has been done. One of the earliest work on SRL on English PropBank(2001) was done by Gildea and Jurafsky (2002). Xue and Palmer (2004) showed that full exploitation of the syntactic tree was needed in the earlier stages to improve the results for semantic role labeling. Towards the recent years, Roth and Woodsend(2014) have shown that vector representation of predicate, arguments and also composition of words leads to improve semantic role labeling.
Since a single system has been made for semantic role labeling for Indian Languages. we take it as the best model and compare our system with them.

4. Semantic Role labeler

Depending on the type of information one wants to learn automatically, there are various ways to construct the semantic role tagging task resting on the annotation of the PropBank of that language. Following the previous work (Anwar and Sharma, 2016) for comparison purposes, we ignore the frameset/word-sense information for now. Therefore we will predict the numbered core arguments ARG[0-3], ARG2 x secondary tags and all ARGM* tags, for each predicate in a sentence. There are also some phrases/chunks in a sentence that are not semantic arguments for predicate in concern and we will label such chunks as NULL. Semantic role labeling can thus be comprehended as a 1 of N classification task but so is not the case. Let us look why in the next section.

4.1. Selecting Approach

As shown in the previous work on Indian Languages (Anwar and Sharma, 2016), direct classification of roles without filtering NULL arguments gave very poor results as compared to the two step approach. In one of the earliest work (Xue and Palmer, 2004), it is observed that for a given predicate, many chunks in the syntactic/dependency tree don't act as its semantic argument. So, the null class count overwhelms the argument count for the given predicate and classifiers will not be efficient in predicting the right argument or classifying them. Also, the features required for checking whether a chunk is an argument or not can be different from the features used to classify roles. Another reason for using this architecture is that it saves a lot of training time for the classifier in the second step. Hence, we follow the 2 step approach, i.e., first identifying the null labels and then classifying the rest. Therefore, we first train a binary classifier to label each chunk as a semantic argument or not. For hindi, the reduces the training data by 51% and for Urdu it is reduced by 81%. Some of the NULL arguments also go to the next step (10% for Hindi and less than 1% for Urdu). Also, some of the non-NULL arguments are filtered out in the first step. Second, we train a multi-class classifier to label the chunks in all classes including the NULL class.

4.2. System Architecture

We do semantic role labeling at a phrase/chunk level. We can break our approach in three major steps along with null data chunk removal as the 0th step.
Step 0: From the dataset we chose, we simply do not take the sentences which have no semantic annotation, i.e., we remove the sentences not having the argument labels and information about the verbs("pbrole annotation") and their frames. If no information is present, we remove the sentence.
Step 1: We run a binary classifier to classify the constituents as Arguments or Non-Arguments(NULL).
Step 2: We run a multi-category classifier to classify the constituents that are labeled as arguments into one of the classes plus NULL.
For the 2nd and final step, we used a Support Vector Machine(SVM) Classifier from the sci-kit library(in Python). Their SVM is a multi-class classifier which learns unique boundaries for each class by taking one vs rest approach for training every class. The classifier's soft boundary can be tuned to maximize results till there is not over-fitting. We also tried our hands using a simple 2 layered neural network having the 1st layer equal to the number of fea-

tures(intuition based) and the last layer equal to the number of classes. We see the outputs from it were also similar which tells us that in our case it largely depends on the features what we give to a machine irrespective of the classifiers. Let us take a look at the features used in previous work and the advancements that could be done.

5. Features

First, we go through the some of the features and techniques used in previous works. We only take the features from the previous baseline that we have used in our system also. We then show the features introduced by us to improve the performance of the system from the current baseline.

5.1. Baseline Features

We take features from the previous system and consider them as baseline for us.

Predicate - predicate word is taken as it is.
Head-word - Head of the chunk/phrase according to syntactic-relations.
Head-word POS - Its Part-Of-Speech category.
Chunk Type - syntactic category (NP, CCP, VGF etc. of the chunk)
Dependency/karaka relation - syntactic relations between chunks.

We look at the use of the above features for both Argument Identification and Argument Classification tasks. The predicate alone cannot tell us any information about identification or labeling but when it is used with other feature such as the head word and head-word's POS, then only it makes sense whether this head word's chunk belongs to a label or not. The head word is an important feature as some of the chunk heads are more likely to be certain arguments for a predicate. This also accounts for the use of predicate as a feature. We use the head-word POS tag along the above because of similar reasons. When used with the predicate, the phrase/chunk tag is useful for identification task because for a predicate, a certain tagged chunk will be more probable to be an argument or a non-argument. The use of karaka relation, a property from the syntactic dependency tree was shown to be one of the best features in this task (Anwar and Sharma, 2016) for Hindi and Urdu. Also, as an inspiration from "Analysis of the Hindi Proposition Bank using Dependency Structure" (Vaidya et al., 2011), we incorporate this feature because of the mappings in their paper show that there is a good interrelationship between the syntactic and semantic predicate-argument relations in a sentence. On account of similar reasons, we also use these features for the argument classification task.

5.2. New Features

After analyzing the PropBanks of both the languages, we came up with certain new features for which we had an intuition that they will contribute significantly towards this task. These are discussed below:

5.2.1. Argument Identification

The following features are added for Argument Classification task as well.

Predicate(verb)'s root form and suffix features - Using the predicate word directly as a feature increases the number of unique instances for the same. To tackle this, we use break the word into its root/stemmed form plus its suffix. This highly reduces the number of distinct verbs for our system as many words fall into the same root category which in-turn gives a boost to our results. For example, in English the predicate 'play' can be present in a sentence as 'playing' or 'played' but both fall under the same predicate. Therefore we take the root 'play' and the suffix 'ing' or 'ed' separately.

Head-Word embedded as a vector - This is the most important feature which contributes in lifting up our results. The reasons to switch over to a new representation are quite similar to the reason for predicate word. In this case, the number of distinct words are a lot more(4300 for Hindi and 4100 for Urdu) because of the richness of the languages used. In a language, the number of verbs(in their root form) are indicatively smaller than the nouns or pronouns. Majority of the chunks we label are of the syntactic category-NP and hence the heads will be nouns or pronouns in a good amount which increases the complexity of this feature. Referring Gensim's Word2Vec, we experimented using their Continuous Bag Of Words approach to create a vector representation of head-word which is of size 30. This highly reduced the size of this feature and improved quality of our results.

Path - It has been shown in many of the previous works that path between the chunk and the predicate has been an important feature in argument identification if not classification. We call for this feature because in Hindi, certain path configurations are more likely to be the arguments to a verb than others. For example, a chunk with path NP↑VGF is more likely to be an argument and a chunk with path JJP↑VGF.
Along with this, we also use dependency path from the intuition that it may further help in classifying the chunks into argument labels according to their syntactic dependence in tree.

Parent and Grandparent's syntactic category - Going through the data and looking at tree structures of the sentences, it is evident that in a good number of cases, either the parent or the grandparent is seen as the predicate for a chunk. Along with the syntactic category we also use the parent-grandparent dependency relation to support the classification.

5.2.2. Argument Classification

The following features were only added for this task.

Chunk's Vibhakti/Post-positional - 'Vibhakti' is a Sanskrit term which is used for post-positions and suffix in Indian languages. In case of Hindi which uses post-positions

instead of prepositions as in English, the post-position similarly provides a good discrimination in selecting the semantic labels.

Other than these we also tried using speech and voice of the predicate chunk. We have not included it in the final system because the results declined adding them to the baseline. Also, there are some chunks in the data which belong to null syntactic categories (Begum et al., 2008). These are namely NULL__NP, NULL__VGF, NULL__CCP. While extracting the features and using them for training, we also appended their non-null categories for every feature where chunk's syntactic category is needed. For example, a NULL__NP chunk is also given the category NP at training time. In the next section, we show the results of these features and compare our system with the existing work.

6. Results and Experiments

We performed our experiments in two phases. The first phase includes experiments on the same train data and test data for both Hindi and Urdu as used in the earlier work (Anwar and Sharma, 2016). The second phase was to make the results more generic over the data and hence we did a 5-fold Cross validation making the train to test ratio as 80% train to 20% test.

Phase 1-

Argument Identification. The results for Hindi and Urdu are shown in table 1 and table 2 respectively. For this step, we trained a binary SVM classifier. We did experiments tuning the hyper-parameters of the classifier and finally got the best results by using - Penalty/Regularization parameter, C = 100.0 and we used a 'rbf' Kernel with kernel coefficient, γ = 0.0005 for Hindi. For Urdu we used similar tuning with C = 80.0 and γ = 0.0006.

System	Precision	Recall	F-Score
Previous Baseline	88	87	87
This work	91.41	90.49	90.94

Table 1: Argument-Identification results for Hindi

System	Precision	Recall	F-Score
Previous Baseline	78	79	78
This work	92.05	91.49	91.76

Table 2: Argument-Identification results for Urdu

Argument Classification. We begin by building the results on baseline features in Hindi shown in table 3. In the next step, we conduct experiments for each of the new feature we propose. This helps us to see the gain and importance of individual feature in the system.

To convert our head-word to vector representation, we used Gensim's Word2Vec tool which is a python library having the Continuous Bag of Words(CBOW) (Mikolov et al., 2013) approach. To train that model on our language, for both Hindi and Urdu we used raw sentences from their corresponding Dependency TreeBanks. For both languages, we used around 200,000 tokens to train the Word2Vec model and used that model for our feature conversion. Table 4 and Table 5 shows the comparison of Argument-Classification(including NULL class) for Hindi and Urdu respectively.

Feature	Precision	Recall	F-Score
Baseline	56.04	49.55	52.59
+Predicate's Root and Morph	60.29	52.88	56.39
+Head-word Vector	61.97	62.12	62.04
+Path	61.15	58.28	59.68
+Parent POS	59.15	55.56	57.29
+Grand-Parent POS	58.33	55.07	56.65
+Vibhakti POS	60.41	59.93	60.16
+Speech and Voice POS	55.32	49.47	52.23

Table 3: Argument-Classification Feature-Wise results for Hindi

System	Precision	Recall	F-Score
Previous Baseline	58	42	49
This work	65.01	66.62	65.80

Table 4: Argument-Classification results for Hindi

System	Precision	Recall	F-Score
Previous Baseline	87	85	86
This work	86.72	86.37	86.54

Table 5: Argument-Classification results for Urdu

Phase 2- In phase 1, the data was split in train and test as shown in Section 2 which is- For Hindi, 100,000 tokens were taken as training data and 20,000 tokens as test data which is a 83.33% train to 16.67% test data; For Urdu, 130,000 tokens as train and 30,000 tokens as test were chosen which is 81.25% train to 18.75% test data. Instead, we came up with a more generic approach which was to split the total data(train+test) used in both Hindi and Urdu to a 80-20 split which can provide for a 5-fold cross-validation. Hence, we present our results averaged after cross-validation as our final results for Hindi and Urdu in table 6 and table 7 respectively. The **Argument-Classification*** results are excluding the NULL class's contribution in the result.

Task	Precision	Recall	F-Score
Argument-Identification	91.08	89.93	90.50
Argument-Classification	64.26	65.90	65.06
Argument-Classification*	69.12	73.16	71.12

Table 6: 5-fold Cross-Validation for Hindi

The previous work did not report the results excluding NULL class making it difficult for comparison purposes since the results would vary dependent on how much NULL

Task	Precision	Recall	F-Score
Argument-Identification	93.79	93.43	93.60
Argument-Classification	84.71	84.89	84.80
Argument-Classification*	85.37	86.27	85.81

Table 7: 5-fold Cross-Validation for Urdu

arguments are present at the classification step. In other way it is dependent on the performance of the Identification task. All experiments for classification step were carried out using SVM classifier with hyper-parameters tuned as - C = 70.0 and we used a 'rbf' Kernel with kernel coefficient, γ = 0.0005 for Hindi. For Urdu we used similar tuning with C = 300.0 and γ = 0.0006.

7. Future Work

This paper does not focus on handling cases where multiple arguments of the same predicate are assigned the same role which is theoretically not possible. To avoid this, we need to use some re-ranking method to assign the best possible set of arguments for a given predicate such that the likelihood of our output is maximized. This can be handled using Integer Linear Programming (Punyakanok et al., 2004)

8. Bibliographical References

Anwar, M. and Sharma, D. (2016). Towards building semantic role labeler for indian languages. In Nicoletta Calzolari (Conference Chair), et al., editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Anwar, M., Bhat, R. A., Sharma, D., Vaidya, A., Palmer, M., and Khan, T. A. (2016). A proposition bank of urdu. In Nicoletta Calzolari (Conference Chair), et al., editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Begum, R., Husain, S., Dhwaj, A., Sharma, D. M., Bai, L., and Sangal, R. (2008). Dependency annotation scheme for indian languages. In *IJCNLP*, pages 721–726.

Bharati, A., Sangal, R., and Sharma, D. M. (2007). Ssf: Shakti standard format guide.

Bhatt, R., Narasimhan, B., Palmer, M., Rambow, O., Sharma, D. M., and Xia, F. (2009). A multi-representational and multi-layered treebank for hindi/urdu. In *Proceedings of the Third Linguistic Annotation Workshop*, ACL-IJCNLP '09, pages 186–189, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christensen, J., Soderland, S., Etzioni, O., et al. (2010). Semantic role labeling for open information extraction. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 52–60. Association for Computational Linguistics.

Koomen, P., Punyakanok, V., Roth, D., and Yih, W.-t. (2005). Generalized inference with multiple semantic role labeling systems. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 181–184. Association for Computational Linguistics.

Liu, D. and Gildea, D. (2010). Semantic role features for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 716–724. Association for Computational Linguistics.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Pizzato, L. A. and Mollá, D. (2008). Indexing on semantic roles for question answering. In *Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering*, pages 74–81. Association for Computational Linguistics.

Pradhan, S., Hacioglu, K., Krugler, V., Ward, W., Martin, J. H., and Jurafsky, D. (2005). Support vector learning for semantic argument classification. *Machine Learning*, 60(1):11–39.

Punyakanok, V., Roth, D., Yih, W.-t., and Zimak, D. (2004). Semantic role labeling via generalized inference over classifiers. Technical report, ILLINOIS UNIV AT URBANA-CHAMPAIGN DEPT OF COMPUTER SCIENCE.

Vaidya, A., Choi, J. D., Palmer, M., and Narasimhan, B. (2011). Analysis of the hindi proposition bank using dependency structure. In *Proceedings of the 5th Linguistic Annotation Workshop*, pages 21–29. Association for Computational Linguistics.

Xue, N. and Palmer, M. (2004). Calibrating features for semantic role labeling. In *EMNLP*, pages 88–94.