Deep Learning methods for Semantic Role Labeling in Indian Languages

Aishwary Gupta, Akshay Pawale and Manish Shrivastava
IIIT Hyderabad, India
{aishwary.gupta, akshay.pawale}@research.iiit.ac.in
m.shrivastava@iiit.ac.in

Abstract

We introduce a supervised deep learning model for Indian languages namely, Hindi and Urdu which uses minimal syntax and yet improves over the current baseline for these languages significantly. We progress with three different models inspired from the recent advancements in this field. In the first model we make use of sequence modeling to generate dependency path embeddings and jointly learning the classification process i.e., Identification and Labeling of arguments. The second model is a syntax-agnostic model where we encode the full sentence using a bi-directional LSTM encoder only using the raw words/tokens. The third and the final model adds dependency label to the previous model making it syntax-aware and performs very well compared to the other models. Finally, we talk about evaluation metrics and analysis of these models.

1 Introduction

Semantic role labeling (SRL) is one of the fundamental tasks in Natural Language Processing (NLP) which aims to automatically learn about the predicate-argument structure for each predicate in given a sentence as the input. It is a type of shallow semantic parsing which labels all the involved constituents in a sentence pertaining to each verb (predicate), answering queries such as who did what to whom, when, and where etc. Semantic roles for a predicate generally are Agent, Patient etc., and adjuncts like Temporal, Locative, Cause etc. Since SRL provides a semantic analysis of a sentence, it can be used in many fields of NLP. Applications of SRL have been shown in topics like information extraction (Bastianelli et al., 2013; Christensen et al., 2010), question-answering (Pizzato and Mollá, 2008; Shen and Lapata, 2007) and machine translation (Liu and Gildea, 2010). SRL has been a trending topic in the research areas of NLP and as a result we have seen great contributions in the form of systems and datasets for various languages. Though much work has been done towards SRL for resource-rich languages like English and Chinese but SRL for Indian Languages (ILs) was pioneered quite recently (Nomani and Sharma, 2016) . Following which another system was recently published (Gupta and Shrivastava, 2018). Both these systems are based on traditional approaches towards SRL as seen in Gildea and Jurafsky (2002) and Xue and Palmer (2004). In this work, we try to apply relatively recent approaches for SRL which include the use of neural networks and find out which works to be the best. Formally, we build three systems-
Model A: Encoding paths between constituents and the predicate using LSTM
Model B: Encoding the whole sentence in a bidirectional LSTM without using any syntax at all.
Model C: Adding a syntactic feature to Model B becoming syntax-aware.

Model A is an extension for the traditional approach where we, (1) first perform Argument Identification i.e., binary classification and (2) then Argument Classification on the probable arguments passed on by step 1. This is often done by training classifiers like SVM on features extracted from the training data. The features are based on syntactic information which is crucial for this approach (Punyakanok et al., 2008). In this model, we find an embedding for the dependency path between a constituent and the current predicate and combine that with other features as used

Proc. of ICON-2018, Patiala, India. December 2018 ©*2018 NLPAI*, pages 59–68

in earlier systems (Roth and Lapata, 2016). This model performs slightly better than the current baseline in ILs (Gupta and Shrivastava, 2018). The gold data is a human-annotated corpus with fully descriptive dependency trees, it has a few chances of error. But in reality we would be given a raw input sentence that would then be parsed by the system and could bring in more error than gold data and affect training badly. Results using automatic parses in the first work (Nomani and Sharma, 2016) verify this. So we apply the recent advancements in SRL which are syntax-agnostic. Model B takes the input as a raw sentence in form of tokens and the only prior information is what are the predicate(s) in this sentence. The sentence is considered as a sequence and we perform SRL as a sequence labeling task. Surprisingly, even for such low resourced language as Hindi and Urdu are, this outperforms Model A and hence the previous baseline. Model C, along with the words adds the dependency label as a feature in the sequence and consequently performs even better than Model B which can be attributed to the use of syntactic feature(s). This makes it the best system available for Hindi and Urdu.

2 Related Work

There are mainly two ways to approach semantic role labeling. The first approach is a traditional one which uses some features extracted out of the gold data. The gold data is a syntactic parsed data with details such as constituent boundaries, syntactic categories, the whole tree structure of the sentence with syntactic relations, part-of-speech(POS) tags for each token in the sentence etc. These features are then used to train a linear classifier like SVM for the tasks of Argument Identification and Argument Classification subsequently (Xue and Palmer, 2004). These tasks can also be done as a single multi-label classification task (Surdeanu and Turmo, 2005). The best predicate-argument structure is chosen at inference stage by techniques like integer linear programming (Punyakanok et al., 2004) or dynamic programming (Täckström et al., 2015). At this stage, structural constraints may lead to improvement in results (Punyakanok et al., 2008). These constraints are either linguistically driven or they exist as a result of the annotation process.

In the last few years, significant work has been done towards fully syntax-agnostic approaches using deep neural networks. Collobert et al. (2011) was the pioneer in introducing such an approach which considers SRL as a sequence labeling task using a convolutional neural network. It takes as input the raw sentence and the constituent boundaries. Although, their approach could not beat the best systems which were still using traditional approaches. The breakthrough in syntax-agnostic SRL was done by Zhou and Xu (2015) whose system differs from Collobert et al. (2011) by using a Deep Bidirectional LSTM network which takes only the predicates indices as input along with the raw sentence. More recently, an end-to-end semantic role labeler was built by He et al. (2017) in which they first identify the predicate(s) in a given input sentence and then for each identified predicate, label the arguments with respective boundaries in the sentence. In such models, the raw sentences tokens are first converted to vector form embeddings taken from pre-trained models. All the above was done for span-based SRL. Marcheggiani et al. (2017) did a similar approach for dependency based SRL. In dependency based SRL, the task it to label the head words of constituents given the gold data has a dependency tree structure. The data for both Hindi and Urdu uniquely have a dependency parsed structure with span-based labeling unlike other languages like English, Chinese etc. where span-based SRL is done for syntactic parses whilst head/dependency-based SRL is done for dependency parses. Before these systems became popular, neural networks were used in syntax-aware systems also (FitzGerald et al., 2015; Täckström et al., 2015; Roth and Lapata, 2016).

The first one to work on SRL for ILs was Nomani and Sharma (2016). They use simple features like dependency labels, syntactic categories, head word of the chunk, head words POS tag, Named entities etc. Gupta and Shrivastava (2018) improved the labeler by introducing features like word embeddings to address the data sparsity issue, path from chunk[1] to predicate and post-positionals of the chunk. These fall under the traditional approaches discussed above. Our first model uses neural method but only to model the dependency path and therefore it still majorly relies on syntactic features. Vaidya et al. (2011) has shown that the predicate-argument structure is

[1]Similar to previous work, we call a word-span/constituent as a chunk.

closely related to dependency relations. The same was proven in the system by Nomani and Sharma (2016) where dependency labels used alone gave good F1-scores for both Hindi and Urdu. Though when they used automatic labels, there is a huge drop in results mainly due to errors in the dependency parse. Dependency/Syntactic parsing in itself is a difficult problem and hence we build a fully syntax-agnostic model to eliminate our reliance on syntax. Although, our third model shows that syntax can still improve performance if it is free from errors.

3 Models

3.1 General Pipeline

The general architecture of SRL is explained in this section. The first step is to identify semantic predicates in the input sentence. In English propbank, there are both verbal and nominal predicates (Hajič et al., 2009), whilst in Hindi and Urdu only verbal predicates are present as of now. Next, the system should disambiguate word-sense of the predicate in consideration. Propbank has multiple senses for verbs and each sense of the same verb can have different labels. This step can be used to improve training or it can just be used at inference time by looking in the corresponding frameset files, the specific roles a verb with given sense can have. The next two steps are Argument Identification and Argument Classification. Argument Identification is done because a high number of candidate arguments have the role **NULL** which may affect the decision boundaries if a classifier is learned directly on full candidates (Xue and Palmer, 2004). Argument Classification is then done to label the remaining candidates which are the most potential arguments from the previous step. For Hindi and Urdu, labeling is done at the chunk/constituent level. A re-ranker using integer linear programming or dynamic programming can be applied as a last step to get the best argument structure for the sentence. For each predicate, we have the identified arguments, each with its scores for every role class. Now we may apply some constraints(structural/linguistic) on the possible output structure and penalize some of the outcomes. Finally we get the result with the best possible predicate-argument structure.

3.2 Sequence Modeling and LSTM

The long short-term memory (LSTM) network is an alternative architecture for a Recurrent neural network (RNN) where each block is a LSTM cell/unit instead of a typical neuron. RNNs process a sequence token by token where each block is given the previous information plus the current token information.

x and y are the input and output respectively, (t) denotes the time step, w_f^m and w_f^m are the matrix from input or recurrent layer to the hidden layer and σ is the activation function. Without $y^{(t-1)}$ term, the RNN model becomes a feed forward network only with number of layers equal to sequence length. RNN is shown in the equation below:

$$y_m^{(t)} = \sigma(\sum_f w_f^m x_f^{(t)} + \sum_i w_i^m y_i^{(t-1)}) \quad (1)$$

When we take one-hot encoded/binary-coded features(as vector), the representation is not effective since data for Hindi and Urdu is quite sparse and vector size is large. To address this, we experiment with recurrent networks to improve the feature representation/encoding and reduce its dimensionality. We use RNNs variant, the LSTM network because it is known to handle long range dependencies (Zhou and Xu, 2015) and in a sentence, the current word is quite dependent on distant words. Also, gradient parameters may vanish or explode especially in processing long sequences (Bengio et al., 1994). To resolve these issues, long short-term memory (Hochreiter and Schmidhuber, 1997) was presented.

LSTM Network. Long short-term memory(LSTM) (Hochreiter and Schmidhuber, 1997; Graves et al., 2009) has a modified architecture with respect to a simple RNN. Memory blocks are used instead of the hidden neural units. The memory block may contain several cells which are activated three multiplicative gates: the input gate, the forget gate and the output gate. These changes improve the RNN model for sequence modeling. Figure 1 shows a basic LSTM cell.

y is the output from memory block. h is the hidden value which equals $'y'$ from the basic RNN model discussed above. c is the cells state value for block m. Number of cells in a block is fixed to be one. α, β and γ stand for the input, forget and output gates activation values. All three mul-

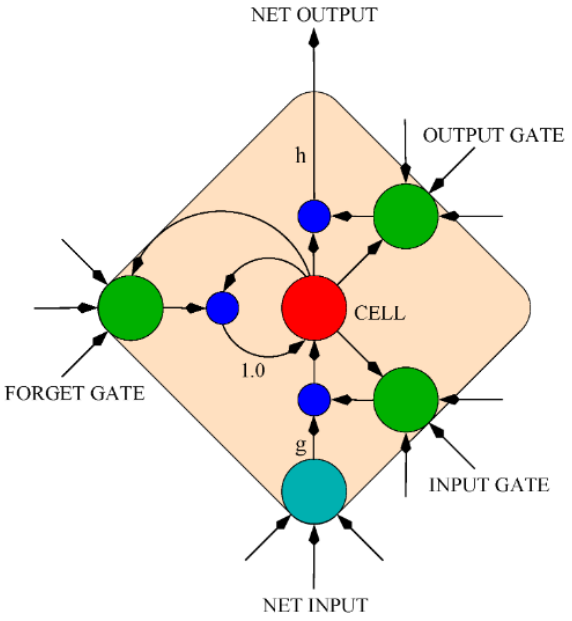

Figure 1: LSTM memory block with one cell. (Graves et al., 2009)

tiplicative gates have different activation σ respectively and the computations are done as follows:

$$
\begin{aligned}
h_m^{(t)} &= \sigma_h(\sum_f w_{f,h}^m x_f^{(t)} + \sum_i w_{i,h}^m y_i^{(t-1)}) \\
\alpha_m^{(t)} &= \sigma_\alpha(\sum_f w_{f,\alpha}^m x_f^{(t)} + \sum_i w_{i,\alpha}^m y_i^{(t-1)} + w_\alpha^m c_m^{(t-1)}) \\
\beta_m^{(t)} &= \sigma_\beta(\sum_f w_{f,\beta}^m x_f^{(t)} + \sum_i w_{i,\beta}^m y_i^{(t-1)} + w_\beta^m c_m^{(t-1)}) \\
\gamma_m^{(t)} &= \sigma_\gamma(\sum_f w_{f,\gamma}^m x_f^{(t)} + \sum_i w_{i,\gamma}^m y_i^{(t-1)} + w_\gamma^m c_m^{(t)})
\end{aligned}
\tag{2}
$$

The gates allow the cells to store and access information over long periods of time/long steps. When the input gate is closed, the new coming input information will not affect the previous cell state. Forget gates remove some historical information over time steps. The output gate should be open for a cell, if rest of the network has to access this cells stored value. In NLP related problems, structural knowledge can be accessed by training the sequences both forward and backward so that the contextual information from left as well as right can be incorporated for better inference. Thus bi-directional LSTM (BiLSTM) was proposed (Schuster and Paliwal, 1997). The BiLSTM which we use is slightly different from their's. We take a LSTM layer to processes the sequence in forward direction whose output is taken by the next LSTM layer as input, where the connections are in backward direction. Pairs of these forward and backward layers can be stacked together to make a Deep BiLSTM proposed in earlier work (Zhou and Xu, 2015).

3.3 Model A - Path embedding model

This model closely follows the architecture of PathLSTM (Roth and Lapata, 2016) and is shown in Figure 2. Given a candidate chunk, first we find the path from this chunk to the predicate being considered and initialize its embedding as follows. Each node in the path is represented as the head-word embedding, POS tag of the chunk, dependency relation with the parent chunk, all three concatenated. Note that this path makes connections at the chunk level only and ends at the predicates chunk. The head-word embeddings are pre-trained embeddings computed similarly as previous work (Gupta and Shrivastava, 2018). The POS tag and dependency relation are given a one-hot vector initialization. Now, we use our sequence model to compute the vector representation of this path.

This differs from the previous work (Gupta and Shrivastava, 2018) as they represent each distinct path as a one-hot encoded vector which is probably not a very optimal way since the number of distinct paths is quite high. For example, in Hindi propbank, even considering only the chunk POS categories[2] in our dependency paths from argument to predicate, nearly 2400 unique paths are present in the training data. If we take the dependency labels only in path (considering direction as argument to predicate), there are around 5900 distinct paths out of which major paths are k1↑root, k2↑root, ccof↑root etc., which occur for only 3.7%, 2.7% and 2.4% respectively (root signifies the predicate). This implies, for almost all paths amongst these, the training data is very less to make any significant improvement in learning from path as a feature. Further, using one-hot implies that we assume that certain paths are likely to impact role labeling in a similar way which may not be true (Roth and Lapata, 2016). Thus, some representation learning should be done, instead of taking the full path as it is.

The dependency path is given to the LSTM

[2]Number of distinct POS categories at chunk level is 12.

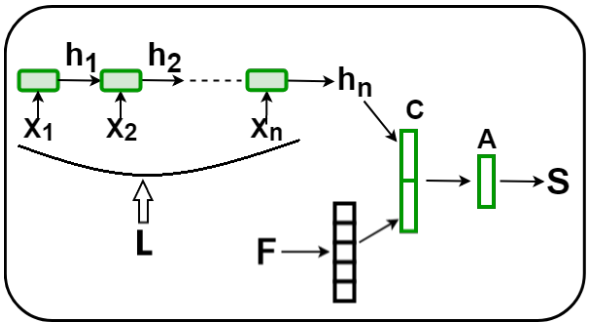

Figure 2: Model A - Path embeddings with LSTM

network as a sequence. The path is taken from the argument chunk to the predicate chunk. Particularly, an element x_i corresponds to the head-word of the chunk w_i, followed by POS category of chunk and then its dependency relation with the next chunk in path, x_{i+1}. The last blocks output state from the network gives us the embedding of this path. As shown in Figure 2, the embedding of a dependency path, specifically is h_n which is returned by the LSTM layers last block after the input of the last element of the sequence, x_n, which corresponds to the initialized encoding of the predicate's chunk.

We use syntactic categories, dependency roles and head-word of chunks in the path because all of them can affect the decision of role labeling (Roth and Lapata, 2016; Gupta and Shrivastava, 2018).

Model A is depicted in Figure 2 and its components are: (1) **L** is the LSTM network which takes input of the length of our path where each node is initialized as discussed above, (2) **F** is the additional input layer which takes features other than the path as input, (3) layer **C** concatenates two neural layers: the upper block is a linear neural layer which takes last node (h_n) of L (dependency path embedding) as input and the lower block is another linear neural layer which takes input from F, (4) layer **A** applies an activation function on the input it receives, and finally (5) **S** is a softmax classifier which produces output for each class $k \in K$ depending on the task(identification or classification). This makes a joint learning model which learns dependency path embeddings and performs SRL as well. Formally, we are given a initialized dependency path X with elements $x_i \in \{x_1, ..., x_n\}$ where n is the length of the path and the features F as one-hot encodings. The LSTM formalization computes hidden embeddings at each step $h_i \in \{h_1, ..., h_n\}$ but we only need the embedding h_n which makes our LSTM network slightly modified than the usual one because gradient parameters will be updated depending on just the last blocks output. We formalize the next layers as follows :

$$C = (W_L h_n + b_L)|(W_F F + b_F)$$

$$A = relu(C)$$

$$S_k = A_k / \sum_{k \in K} A_k$$

We perform the training for argument identification and argument classification separately following the findings from earlier work in English (Xue and Palmer, 2004) as well as for Hindi and Urdu (Nomani and Sharma, 2016). This also means that different path embeddings are learned depending on what the task is. The features F are taken as it is from the current baseline (Gupta and Shrivastava, 2018) for ILs for making our comparison more obvious. These features are - predicate word(verb) root form, its suffix separately, head word of the candidate chunk taken from pre-trained embeddings, candidate chunks vibhakti(post-positional), head word POS tag, candidate chunks POS category.

3.4 Model B - Syntax-agnostic deep model

This model takes the sentence as a sequence processed word by word. A sequence say of length L is processed n_p times if the number of predicates in the sentence is n_p. Hence, the time complexity of this model is $O(n_p L)$. At each step in the sequence, the current words embedding and a binary bit indicating whether word is itself the predicate or not, is given as the input. These are the only features needed to train our network.

Given a sentence-predicate pair (s, v) as the input, we have to predict the output sequence y. We have used IOB tagging (Collobert et al., 2011; Zhou and Xu, 2015) for this problem. Therefore, each $y_i \in y$ should belong to the set of IOB tags. The set contains the tags, O - is given to words outside the argument chunk, B_k - is given to words at beginning of the chunks and I_k - is given to the words inside the chunks. k denotes the various roles shown in section 4. Let the length of the sequence be n, where $n = |s| = |y|$. Our goal is to find the highest scoring tag sequence y from all the possible tag sequence for an input. Our model uses a Bidirectional LSTM (BiLSTM) network to

learn a locally decomposed scoring function determined by the input: $\sum_{t=1}^{n} \textbf{log}p(y_t|s)$. To incorporate constraints like IOB order, structural constraints(explained later in this section), we extend the scoring function (He et al., 2017) with penalization terms:

$$f(s,y) = \sum_{t=1}^{n} logp(y_t|s) - \sum_{c \in C} c(s, y_{1:t})$$

Given the input s and length-t prefix $y_{1:t}$, each constraint function c applies a non-negative penalty on the scoring function f.

The model is depicted in Figure 3. The input colored as red is the raw word, binary bit b_i to indicate whether word w_i is the predicate. The dotted box next to it is shown to incorporate additional features if required, which is basically our third model and not of use in this model. The input is then embedded as the concatenation of word embedding and the binary bit at the embedding layer. The next layer is the beginning of our BiLSTM network. Layer L_k is forward if k is odd and backward if k is even. We chose the number of layers as two for reasons given in section 6. Recent best works (Zhou and Xu, 2015; Marcheggiani et al., 2017; He et al., 2017) in English SRL have used up till 8 layers. Finally, the output of this goes to a softmax layer to compute a locally normalized distribution over the output tags.

Constrained Decoding. The approach above does not yet apply any constraints on the output structure. We use A search over tag prefixes to apply constraints on the output structure at decoding time (He et al., 2017). We list some example constraints as follows:

IOB Constraints: We need to ensure that our system does not produce invalid IOB tagging such as B_k tag followed by I_k tag or say I-ARG0 followed by B-ARG1 etc. We apply infinitely high penalty for such transitions.

SRL Constraints: Though there have been no constraints described in the data or in previous work for Hindi and Urdu propbanks, but some structural constraints have been applied for English SRL (Punyakanok et al., 2008; Täckström et al., 2015). We only experiment with the Unique core roles constraint, i.e., numbered arguments (ARG-0,1,2,3) can occur at most once for each predicate.

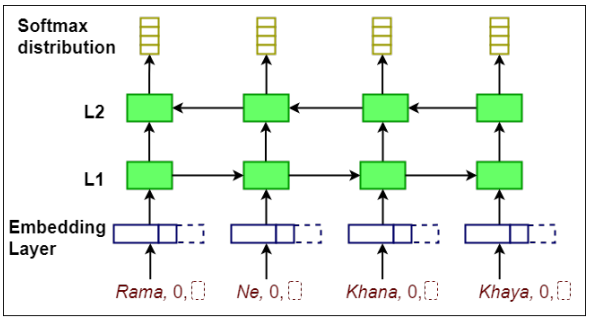

Figure 3: Model B and C - 2 Layer BiLSTM model.

3.5 Model C - Syntax-aware deep model

We build a third model to see the effect of adding syntax to Model B which is fully syntax agnostic. Going with the findings by Nomani and Sharma (2016) that dependency label alone gives great results, we go ahead with it as the only feature to be used. In Figure 3, the red colored input with dotted box is meant to incorporate any other features and this is where we add dependency label. Since the dependency parsing in Hindi and Urdu propbanks is only till the chunk level, so a whole chunk, i.e., all tokens inside are assigned the same dependency relation label.

Formally, at the embedding layer, we now concatenate embeddings of the word, the binary bit indicating if this word is the predicate and the dependency role encoded as a one-hot vector. After this, the training is exactly same as model B.

4 Dataset

We carry out our experiments on Hindi Propbank and Urdu Propbank exactly on the sections used in earlier work. We use the same train and test set for both the languages. Hindi and Urdu propbank are still in the process of annotation. These are build on top of the respective treebanks which has dependency parsed trees with parsing up till the chunk level. The semantic label annotation is also done at chunk level. Chunk boundaries, POS categories of tokens as well as the whole chunk, morphology features etc. are already annotated in the gold corpus. The treebanks and hence the propbanks (since it is build on treebank) of both languages are represented in the Shakti Standard Format (Bharati et al., 2007). Gupta and Shrivastava (2018) did a 5-fold cross validation on this data and the results showed a very slight change with

respect to the train-test split used earlier (Nomani and Sharma, 2016). This shows that the data distribution in the train-test splits is normalized and hence, we don't perform cross-validation on our models in this work. They have also given a good explanation about the data set but they missed to give out any statistics about the data. Therefore, we decided to provide full details on the data with this work and show them in Table 1. The exact data file names used for training and testing are listed in section 8 of this paper.

	Hindi train	Hindi test	Urdu train	Urdu test
Sentences	1643	448	4657	1234
Tokens	36690	9827	133058	35532
Final Sentences	1300	358	988	309
Final Tokens	30141	8285	33374	10628
Propositions	2309	631	1192	391
Verbs	166	94	40	24
Arguments	5872	1620	3745	1207
ARG0	1185	320	433	137
ARG1	2046	559	624	210
ARG2	175	33	99	27
ARG3	4	0	14	2
ARG2-ATR	352	77	53	8
ARG2-GOL	61	13	4	9
ARG2-LOC	54	11	137	54
ARG2-SOU	46	14	79	33
ARGM-LOC	679	210	399	136
ARGM-MNR	350	106	67	21
ARGM-TMP	328	97	210	72
ARGM-ADV	131	38	194	52
ARGM-PRP	114	31	102	26
ARGM-DIS	94	34	25	10
ARGM-EXT	92	23	10	4
ARGM-CAU	83	29	46	4
ARGM-MNS	42	13	24	7
ARGM-DIR	20	8	6	2
ARGM-NEG	7	2	13	1
ARGM-PRX	2	1	1194	393
ARGM-VLV	0	0	0	0
ARGM-MOD	2	0	1	0
ARGA	0	0	1	1
ARGC	0	0	0	0

Table 1: Hindi and Urdu Propbank data statistics.

Final Sentences and Final tokens signify the numbers after filtering out sentences with no predicate. The total number of distinct verbs in full Hindi and Urdu datasets after filtering are 186 and 46.

5 Experiments

Evaluation metrics. To compare Model A with the previous works, we have used the same evaluation metric as used by them, which is to give the results of the tasks of identification and classification separately. Since model B and C do not perform identification and classification separately, we cannot compare this directly to results of previous works and model A. We propose the use of the evaluation script used for Carreras and Màrquez (2005). It gives a more perceivable understanding of the results of a model since it compares the actual sentences, one-to-one from the gold corpus and from the predictions. According to the script, a prediction is counted correct if and only if it has the exact same chunk boundary and the same label. We also project the results of Model A to test data sentences and evaluate it with conll 2005 script to make a clear comparison of all models we built.

5.1 Model A

We train different networks for argument identification and classification using the PyTorch library for deep neural networks. For direct comparison with previous work, features other than path are same as the ones used in previous work (Gupta and Shrivastava, 2018) and evaluation metric is also the same. The hyper-parameters of the model are as follows:

Learning rate for identification is 0.001 and for classification is 0.0001. Dropout rate is 0 for both the tasks and hidden layer size of LSTM network is 100 for both the tasks. Layer C is composed of two neural blocks each of size 100, whose output of size 200 is sent over to layer A after concatenation. We have used Cross Entropy function to compute loss and Stochastic Gradient descent (SGD) for optimizations. The model was run for 200 iterations for identification task and 300 iterations for classification task. The results for Hindi and Urdu are given in comparison with their corresponding state-of-the-art in Table 2 and Table 3 for identification and classification respectively.

Language	Model	Precision	Recall	F1-score
Hindi	Gupta et. al	91.41	90.49	90.94
	Our Model	**93.35**	**93.29**	**93.32**
Urdu	**Gupta et. al**	**92.05**	**91.49**	**91.76**
	Our Model	91.50	91.17	91.33

Table 2: Argument Identification results.

Language	Model	Precision	Recall	F1-score
Hindi	Gupta et. al	65.04	66.62	65.80
	Our Model	**70.23**	**72.25**	**71.22**
Urdu	**Gupta et. al**	**86.72**	**86.37**	**86.54**
	Our Model	85.57	85.52	84.73

Table 3: Argument Classification results.

5.2 Model B and C

Our BiLSTM has only 2 layers - 1 forward LSTM and 1 backward LSTM. The hidden dimension is set to be 300. A softmax layer for predicts the output distribution. All weight matrices of the BiLSTM are initialized as random orthonormal matrices as described in Saxe et al. (2013). The pre-trained embeddings for both Hindi and Urdu are taken from Fast-Text (Bojanowski et al., 2017). Tokens that are not present in the pre-trained model are given a randomly initialized embedding. Size of the embedding is 300 for both languages.

Training: We use Adadelta (Zeiler, 2012) as optimizer with $\epsilon = 1e^6$ and $\rho = 0.95$ which are also the default parameters available in the proposed paper. We use mini-batches of size 50. We clip gradients with norm larger than 5 and set the RNN-dropout probability to 0.1. All the models are trained for 300 iterations without any early stopping since we dont have a development data to check the development loss. In the final model we only use the IOB constraints.

6 Results and Analysis

The problem with Model A is that it also depends on a feature template which can be language/data specific. Hindi propbank was created automatically while Urdu was purely human-annotated, this is why Urdu's dataset is better and thus it gives better results. The Urdu verbs are also very less in number as seen in Section 4 and thus the system doesn't have to learn a lot of predicates. Also, the majority arguments in Urdu belong to ARGM-PRX which is a complex noun-verb/adjective-verb predicate but this class achieves the best F-score and contributes heavily towards the results. This also means that the data sparsity is low.

In Model A, We chose the path configuration from argument chunk to predicate chunk because it gave better results than the reverse path which is also the actual dependency path. The first model learns the path embeddings and performs only slightly better than the previous work (Gupta and Shrivastava, 2018). The difference in this model and previous work is only that it uses LSTM to encode path while the previous work used a one-hot encoding for this feature. Though, in our case it provides only a slight gain. This can be attributed to the fact that the data available for each unique path is very less to learn a reliable embedding.

In Model B and C, number of layers is kept as 2 because increasing number of layers degraded performance. The primary reason for this could be attributed to less training points in data. We only use IOB constraints since they gave a significant improvement in results whilst the Unique Core Roles constraint did not. These models are overall the best performers as they are in a way learning more complex features from the whole sentence than the pre-defined features used in model A.

Language	Model	Precision	Recall	F1-score
Hindi	Gupta et al.	42.52	49.66	45.81
	Model A	44.38	50.53	47.26
	Model B	56.71	56.39	56.55
	Model C	**70.41**	**70.41**	**70.41**
Urdu	Gupta et al.	86.41	67.16	75.58
	Model A	85.01	66.91	74.88
	Model B	63.10	63.20	63.15
	Model C	**77.98**	**78.16**	**78.07**

Table 4: SRL full task results evaluated using conll-2005 shared task evaluation script.

7 Conclusion and Future Work

More data for both the languages can be helpful to improve the performance further and get a better analysis. A detailed report can then be achieved on what quantity of data is actually required for SRL in low-resource settings. Once more data is available, analysis of results will give the reasons why and how deep learning models perform better than the traditional ones. Results of our paper have shown that deep learning is performing very well even in such low data. This shows that more complex features have been missed when manually extracting features from a parsed sentence. Hence, better feature engineering also lies in the future scope of SRL for Indian Languages.

8 Data Sections

The names of the files in training set and testing set of Hindi are provided in the following link. https://github.com/ashg1910/indian_srl

References

E. Bastianelli, G. Castellucci, D. Croce, and R. Basili. Textual inference and meaning representation in human robot interaction. In *Proceedings of the Joint Symposium on Semantic Processing. Textual Inference and Structures in Corpora*, pages 65–69, 2013.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

A. Bharati, R. Sangal, and D. M. Sharma. Ssf: Shakti standard format guide. *Language Technologies Research Centre, International Institute of Information Technology, Hyderabad, India*, pages 1–25, 2007.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. ISSN 2307-387X.

X. Carreras and L. Màrquez. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the ninth conference on computational natural language learning*, pages 152–164. Association for Computational Linguistics, 2005.

J. Christensen, S. Soderland, O. Etzioni, et al. 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, 2010.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

N. FitzGerald, O. Täckström, K. Ganchev, and D. Das. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 960–970, 2015.

D. Gildea and D. Jurafsky. Automatic labeling of semantic roles. *Computational linguistics*, 28(3):245–288, 2002.

A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868, 2009.

A. Gupta and M. Shrivastava. Enhancing semantic role labeling in hindi and urdu. In K. Shirai, editor, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Paris, France, may 2018. European Language Resources Association (ELRA). ISBN 979-10-95546-24-5.

J. Hajič, M. Ciaramita, R. Johansson, D. Kawahara, M. A. Martí, L. Màrquez, A. Meyers, J. Nivre, S. Padó, J. Štěpánek, et al. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18. Association for Computational Linguistics, 2009.

L. He, K. Lee, M. Lewis, and L. Zettlemoyer. Deep semantic role labeling: What works and whats next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 473–483, 2017.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

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

D. Marcheggiani, A. Frolov, and I. Titov. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. *arXiv preprint arXiv:1701.02593*, 2017.

M. A. Nomani and D. M. Sharma. Towards building semantic role labeler for indian languages. *FC Kochi center on Intelligent Systems (KCIS), IIIT-Hrdrabad, India*, 2016.

L. A. Pizzato and D. Mollá. 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, 2008.

V. Punyakanok, D. Roth, W.-t. Yih, and D. Zimak. Semantic role labeling via integer linear programming inference. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1346. Association for Computational Linguistics, 2004.

V. Punyakanok, D. Roth, and W.-t. Yih. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2): 257–287, 2008.

M. Roth and M. Lapata. Neural semantic role labeling with dependency path embeddings. *arXiv preprint arXiv:1605.07515*, 2016.

A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

D. Shen and M. Lapata. Using semantic roles to improve question answering. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007.

M. Surdeanu and J. Turmo. Semantic role labeling using complete syntactic analysis. In *Proceedings of the Ninth Conference on Computational Natural*

Language Learning, pages 221–224. Association for Computational Linguistics, 2005.

O. Täckström, K. Ganchev, and D. Das. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41, 2015.

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

N. Xue and M. Palmer. Calibrating features for semantic role labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 2004.

M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

J. Zhou and W. Xu. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1127–1137, 2015.