Semantic Role Labelling For Hindi

Authors - Rahul Mehta and Tejasvi Chebrolu, IIIT Hyderabad

Semantic Role Labelling

Semantic Role Labelling, in natural language processing, is a process that assigns labels to different words in a sentence that indicate their **semantic role** in the sentence. This helps in finding the meaning of the sentence, and more importantly, the role of a particular word in creating that meaning of the sentence. The task essentially boils down to identifying the various arguments associated with the *predicate* or the main verb of the sentence and assigning them specific roles.

Semantic role labelling is mostly used for machines to understand the roles of words within sentences. This benefits applications similar to Natural Language Processing programs that need to understand not just the words of languages, but how they can be used in varying sentences.[1] A better understanding of semantic role labelling could lead to advancements in question answering, information extraction, automatic text summarization, text data mining, and speech recognition.[2]

Problem Classification

The problem can then be further decomposed into subproblems. Each of these subproblems are also challenging to solve for a language like Hindi because it is a free order language. This means that the syntax of the language is not very strict and there are very few restrictions on how a sentence must be structured in Hindi. The subproblems are -

Predicate Detection: The first step is finding the predicate in a given sentence.

Predicate Sense Disambiguation: The second step is disambiguating the sense of the predicate found.

Argument Identification: The last step is Identifying the arguments for the given predicate for the given sense.

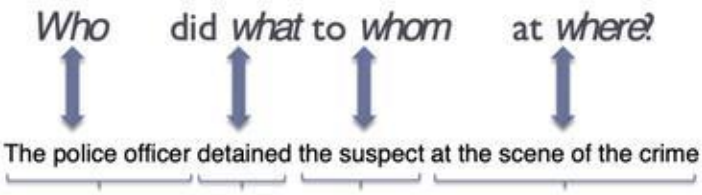
---

[1] Park, Jaehui (2019). "Selectively Connected Self-Attentions for Semantic Role Labeling". *Applied Sciences*. **9** (8) – via ProQuest.
[2] Gildea, Daniel; Jurafsky, Daniel. "Automatic Labeling of Semantic Roles" (PDF). *Association for Computational Linguistics*. **28** (3).

Argument Classification: The final step that completes the process is assigning the labels to the arguments found. **In this project work, we will be focussing on solving this problem of argument classification.**

Example

English



The above example has 3 distinct labels that can be seen - **Agent**, **Theme**, and **Location**. It also has the **Predicate** labelled. Clearly, in this example, the police officer (**Agent**) is the person detaining (**Predicate**) the suspect (**Theme**) at the scene of the crime (**Location**). Using these labels we are then able to answer the question *"Who did what to whom where?"*

Some of the more common labels are -

- Agent
- Experiencer
- Theme
- Result
- Location

More labels (not exhaustive) can be found in these slides. These labels are used for English but for Hindi, the problem is classified in a different way, and hence, the labels are also slightly different. However, the core premise is still the same.

Hindi

There have already been several efforts into making an SRL for Hindi such as "*Towards Building Semantic Role Labeler for Indian Languages*" (Anwar and Sharma, 2016) and "*Enhancing Semantic Role Labeling in Hindi and Urdu*" (Gupta and Shrivastava, 2018). Both of these implementations use a 2 step approach, that is, 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 the second step into the role labels. This is because, for a given predicate, many constituents in a syntactic tree are not its semantic arguments (Xue and Palmer, 2004). So, the non-argument count

overwhelms the argument count for the given predicate and classifiers will not be efficient in predicting the right argument or in classifying them.

For Semantic Role Labelling in Hindi, we will be labelling the words into the following roles:

Label	**Description**
ARG0	Agent, Experiencer, or doer
ARG1	Patient or Theme
ARG2	Beneficiary
ARG3	Instrument
ARG2-ATR	Attribute or Quality
ARG2-LOC	Physical Location
ARG2-GOL	Goal
ARG2-SOU	Source
ARGM-PRX	Noun-Verb Construction
ARGM-ADV	Adverb
ARGM-DIR	Direction
ARGM-EXT	Extent or Comparision
ARGM-MNR	Manner
ARGM-PRP	Purpose
ARGM-DIS	Discourse
ARGM-LOC	Abstract Location
ARGM-MNS	Means
ARGM-NEG	Negation

ARGM-TMP	Time
ARGM-CAU	Cause or Reason

The labels have been taken from this paper (Bhatia et al., 2013). This paper serves as the guideline for the **Hindi PropBank,** which is also used as the dataset for this project.

a. *[ARG0 rAma ne] [ARG1 **darvAzA**] KolA*
Ram Erg door opened
'Raam opened the door.'
b. *[ARG1 **darvAzA**] KulA*
door opened
'The door opened.'

In the above examples, the chunks and their arguments can clearly be seen. In the first sentence, Ram is clearly the ARG0 or the agent as he is the one doing the work. Similarly, n matching the labels with the description of the label provides us with an understanding of the semantics of the sentence.

Dataset

The dataset used for this project is modelled off the Hindi Urdu Propbank. It is a multi-layered treebank for Hindi and Urdu that has multiple features that make creating models for SRLand other NLP tasks easier.

Hindi Propbank

PropBank is a large annotated corpus consisting of information regarding the argument structure of predicates. PropBanking involves creating a semantic layer of annotation that adds predicate-argument structure to syntactic representations. The Hindi PropBank annotations are done on top of the Hindi dependency treebank. For each verb, PropBank represents the information about the arguments that appear with the verb in its corresponding frame file. The arguments of the verbs are labelled using a small set of numbered arguments, e.g. *Arg0*, *Arg1*, *Arg2*, etc.

The consistency in semantic role labelling is also helpful in the training of machine learning systems such as automatic semantic role labellers. However, consistency in semantic role labelling for the arguments of the same verb does not mean that only one set of semantic role (SR) labels is available for a specific verb. PropBank also takes into account the different

senses of the same verb while annotating the semantic roles. It is possible that each of the two senses of the same verb has a different set of SR labels.

ADVANTAGES OF USING THE PROPBANK

- ★ **Dependency Treebank** - The propbank also provides us with the option of looking at the syntax of the sentence. This is done by providing us with a dependency parse of the input sentence. This is very useful as a lot of time, the semantics of a sentence is decided by the syntax of the sentence, especially in Hindi, a free order language.
- ★ **Phrase Structure Treebank** - To get a better understanding of the syntax of the sentence, the propbank also provides us with the phrase structure of the sentence. This is extremely important in SRL as it helps us create the chunks easily while ensuring that no semantic information is lost. A regular chunker would not have been able to do that.
- ★ **Multi-Featured Propbank** - The framefiles in the propbank have been adapted to include morphological causatives, unaccusative verbs, and experiencers. All of these would help increase the accuracy of the SRL.

Dataset Representation

The dataset finally used can be seen in *interim.txt*. Each word in the dataset is a vector of size **308**. This consists of word embeddings of size **300**. These embeddings are important as they help in the neural network as well as help capture more information about the sense of the word that just the word in text would not be able to. The remaining columns are the **Label, Chunk, Postposition, Head-POS, Dependency-Head, Dependency, SRL,** and the **Predicate.** The label is either a 0 or a 1 depending on whether it is a root or not, The chunk gives the POS tag of the chunk. The postposition gives the postposition or whether or not it exists. This is important in Hindi since it is a postpositional language. The Head-POS gives the POS of the head. The Dependency-Head gives the dependency head of that chunk. The Dependency gives the dependency of the chunk on the head. The SRL finally gives us the SRL of the chunk. The Predicate gives us which predicate the chunk is for.

The image shows the head of the data:

```
      dim_0     dim_1     dim_2     dim_3     dim_4     dim_5     dim_6  \
0 -0.005985  0.015627  0.007076  0.006575  0.014297  0.007125 -0.000785
1  0.021908 -0.084995 -0.011690 -0.048774 -0.052874 -0.018173  0.058139
2  0.014475 -0.069880 -0.003023  0.022581 -0.021881  0.012437  0.103112
3 -0.019438 -0.003835 -0.027385  0.019167 -0.001544  0.027654 -0.007453
4  0.000870  0.018253  0.025201 -0.015667  0.002589 -0.026993 -0.012401

      dim_7     dim_8     dim_9  ...    dim_298   dim_299  Label      chunk  \
0 -0.009658 -0.001066 -0.043356  ...   0.008799 -0.023102      1        JJP
1 -0.065300 -0.007800 -0.011115  ...   0.021349  0.001630      0        VGF
2 -0.104538 -0.024584  0.046685  ... -0.050404 -0.116184      0        CCP
3 -0.041042 -0.011774 -0.011934  ...   0.024064 -0.017355      1         NP
4 -0.034212  0.000570  0.025318  ...   0.004674  0.013638      0  NULL__NP

   postposition  head-POS  dependency-head  dependency       srl  predicate
0             0       adj              VGF          k1s  ARG2-ATR      VGF.1
1            है         v                0        root         0          0
2             0       avy          NULL__NP          rs         0          0
3          0_को         n             VGNF         k7t  ARGM-TMP      VGNF
4             0         0                0        root         0          0

[5 rows x 308 columns]
```

Machine Learning Modelling

We train a set of 3 models on a train test split of 66.66% and 33.33% on the available data. We have 23 set of argument classes to predict, so we try to solve the problem of **multi-class argument classification.**

1. Logistic Regression

We train a logistic regression model with l2 regularization on our Hindi dataset and get the following results on the multiclass classification task

classifier_report_logistic

	precision	recall	f1-score	support
0	0.6548067860508954	0.9363207547169812	0.7706600110926235	2968.0
ARG-UNDEF	0.0	0.0	0.0	1.0
ARG0	0.4513888888888889	0.19938650306748465	0.276595744680851	326.0
ARG1	0.5640243902439024	0.3003246753246753	0.39194915254237284	616.0
ARG2	0.0	0.0	0.0	59.0
ARG2-ATR	0.4	0.03333333333333333	0.06153846153846154	120.0
ARG2-GOL	0.0	0.0	0.0	23.0
ARG2-LOC	0.0	0.0	0.0	18.0
ARG2-SOU	0.0	0.0	0.0	12.0
ARG3	0.0	0.0	0.0	2.0
ARGM-ADV	0.0	0.0	0.0	48.0
ARGM-CAU	0.8	0.18181818181818182	0.2962962962962963	22.0
ARGM-DIR	0.0	0.0	0.0	8.0
ARGM-DIS	0.0	0.0	0.0	25.0
ARGM-EXT	0.5	0.030303030303030304	0.05714285714285715	33.0
ARGM-LOC	0.30303030303030304	0.046511627906976744	0.08064516129032258	215.0
ARGM-MNR	0.2	0.00819672131147541	0.015748031496062992	122.0
ARGM-MNS	0.0	0.0	0.0	11.0
ARGM-NEG	0.0	0.0	0.0	4.0
ARGM-PRP	0.0	0.0	0.0	47.0
ARGM-PRX	0.0	0.0	0.0	1.0
ARGM-TMP	0.5666666666666667	0.14166666666666666	0.22666666666666666	120.0
accuracy	0.6386169548010832	0.6386169548010832	0.6386169548010832	0.6386169548010832
macro avg	0.2018144106763935	0.08535734065676387	0.0989655628521143	4801.0
weighted avg	0.5577400246445728	0.6386169548010832	0.5584625330897215	4801.0

2. LSTM

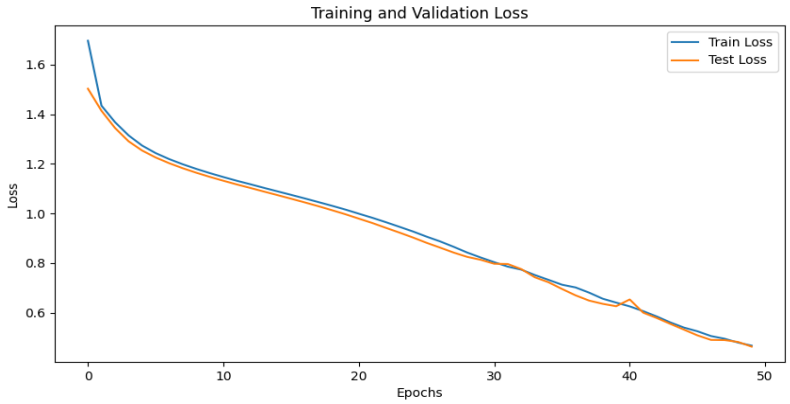
We train an LSTM model with the following hyperparameters and for 50 epochs

```
--EMBEDDING_DIM=300
--NUM_HIDDEN_NODES=100
--epochs=50
--batchsize=64
--learning_rate=0.001
```

Loss - CrossEntropyLoss
Optimizer - SGD with momentum=0.9

LSTM Neural Network Structure - `SRL_LSTM(`

```
  (lstm): LSTM(300, 100, batch_first=True)
  (fc): Linear(in_features=100, out_features=23, bias=True))
```



classification_report_lstm_50e

	precision	recall	f1-score	support
0.0	0.890021624961384	0.9706873315363881	0.9286059629331184	2968.0
1.0	0.0	0.0	0.0	1.0
2.0	0.8322981366459627	0.8220858895705522	0.8271604938271604	326.0
3.0	0.8459167950693375	0.8912337662337663	0.8679841897233203	616.0
4.0	0.72	0.3050847457627119	0.42857142857142855	59.0
5.0	0.8588235294117647	0.6083333333333333	0.7121951219512195	120.0
6.0	0.6666666666666666	0.08695652173913043	0.15384615384615383	23.0
7.0	1.0	0.05555555555555555	0.10526315789473684	18.0
8.0	0.0	0.0	0.0	12.0
9.0	0.0	0.0	0.0	2.0
10.0	0.5714285714285714	0.3333333333333333	0.4210526315789474	48.0
11.0	0.6923076923076923	0.4090909090909091	0.5142857142857142	22.0
12.0	1.0	125	0.2222222222222222	8.0
13.0	0.6666666666666666	0.24	0.3529411764705882	25.0
14.0	0.6666666666666666	0.24242424242424243	0.3555555555555555	33.0
15.0	0.6994818652849741	0.627906976744186	0.6617647058823529	215.0
16.0	0.7058823529411765	0.4918032786885246	0.5797101449275363	122.0
17.0	0.0	0.0	0.0	11.0
19.0	1.0	0.25	0.4	4.0
20.0	0.7777777777777778	0.2978723404255319	0.43076923076923074	47.0
21.0	0.0	0.0	0.0	1.0
22.0	0.7563025210084033	0.75	0.7531380753138075	120.0
accuracy	0.8606540304103312	0.8606540304103312	0.8606540304103312	0.8606540304103312
macro avg	0.6068291303107748	0.3412440102017348	0.3961393620796860 3	4801.0
weighted avg	0.8477430517715253	0.8606540304103312	0.8447622645335907	4801.0

3. Bidirectional LSTM

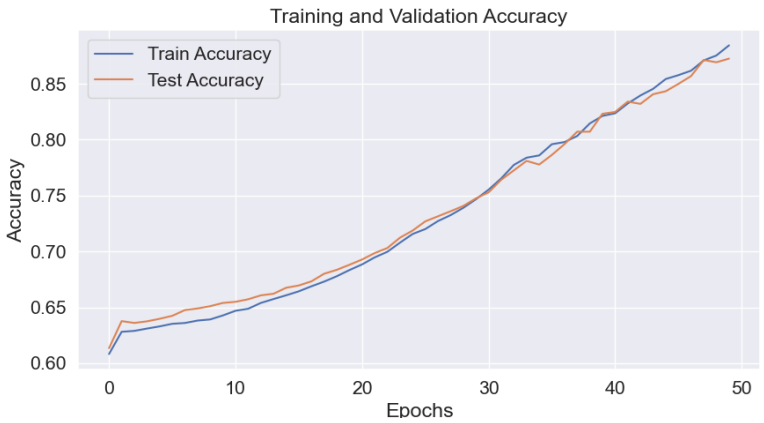
We train a bidirectional LSTM model with the following hyperparameters and for 50 epochs

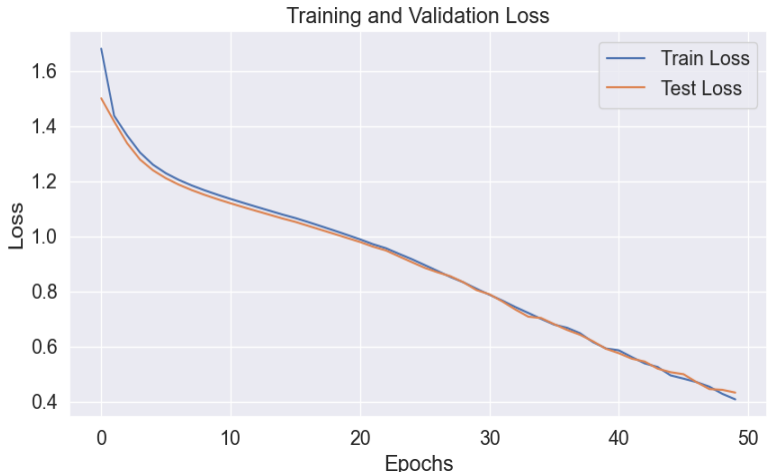
```
--EMBEDDING_DIM=300
--NUM_HIDDEN_NODES=100
--epochs=50
--batchsize=64
--learning_rate=0.001
```

Loss - CrossEntropyLoss
Optimizer - SGD with momentum=0.9

LSTM Neural Network Structure - `SRL_LSTM(`

```
  (lstm): LSTM(300, 100, batch_first=True)
  (fc): Linear(in_features=100, out_features=23, bias=True)
)
```





classification_report_bilstm_50e

	precision	recall	f1-score	support
0.0	0.9033468877072255	0.9730458221024259	0.9369018653690186	2968.0
1.0	0.0	0.0	0.0	1.0
2.0	0.7976539589442815	0.8343558282208589	0.815592203898051	326.0
3.0	0.8480243161094225	0.9058441558441559	0.8759811616954475	616.0
4.0	0.8235294117647058	0.23728813559322035	0.368421052631579	59.0
5.0	0.8791208791208791	0.6666666666666666	0.7582938388625592	120.0
6.0	1.0	0.043478260869565216	0.08333333333333333	23.0
7.0	1.0	0.16666666666666666	0.2857142857142857	18.0
8.0	1.0	0.08333333333333333	0.15384615384615385	12.0
9.0	0.0	0.0	0.0	2.0
10.0	0.6538461538461539	0.3541666666666667	0.4594594594594595	48.0
11.0	0.8181818181818182	0.4090909090909091	0.5454545454545455	22.0
12.0	1.0	125	0.2222222222222222	8.0
13.0	0.8888888888888888	0.32	0.47058823529411764	25.0
14.0	0.6666666666666666	0.36363636363636365	0.4705882352941177	33.0
15.0	0.7028301886792453	0.6930232558139535	0.6978922716627635	215.0
16.0	0.7472527472527473	0.5573770491803278	0.6384976525821596	122.0
17.0	0.0	0.0	0.0	11.0
19.0	0.0	0.0	0.0	4.0
20.0	0.7857142857142857	0.23404255319148937	0.36065573770491804	47.0
21.0	0.0	0.0	0.0	1.0
22.0	0.8165137614678899	0.7416666666666667	0.777292576419214	120.0
accuracy	0.8708602374505311	0.8708602374505311	0.8708602374505311	0.8708602374505311
macro avg	0.6514349983792823	0.3503946515246941	0.40548794688381573	4801.0
weighted avg	0.8642834532358544	0.8708602374505311	0.8558936474190171	4801.0

4. Summary

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	63%	56%	64%	56%
LSTM	86%	84.7%	86%	84.4%
Bidirectional-LSTM	87%	86.4%	87%	85.5%

We conclude that the Bidirectional LSTM is the best model for the SRL Task on Hindi Propbank.

Codes

1. Github Repository - https://github.com/rahcode7/SRL

2. Final Presentation - https://docs.google.com/presentation/d/1SDtWxaUdr3IW4LmF2HdAjiCDr1s5f38EhBQy0dvsNug/edit?usp=sharing