Semantic Role Labelling For Hindi

Team Logical Parsers

Satya G. - 2023202015
Hrishikesh N. - 2023201012
Nikith D. - 2023201057

Semantic Role Labelling

- Semantic Role Labelling (SRL) is a task in NLP that assigns semantic roles to words and phrases in a sentence. It indicates the relationship between words and the main predicate or verb.
- SRL extracts information about who did what to whom, when, where, and how in a given sentence.

Semantic Role Labelling - Cont.

- SRL identifies different roles played by various elements of a sentence corresponding to the action given by the verb. It plays a key role in understanding sentences and the role of specific words in a given sentence.
- Development of better SRL methods will lead to improvements in machine translation, text summarization, text generation, sentiment analysis, question answering, and information extraction.

Semantic Role Labelling - subproblems

SRL can be broken down into different subproblems:

- Predicate Detection: Identifying the predicate in a given statement.
- Argument Identification: Identifying the arguments for the given predicate in the given sentence.
- Role Labeling: Assigning roles to the arguments found.

Each one of these is a challenge in free-order languages like Hindi. So, our work mainly focuses on Role Labeling for SRL in Hindi.

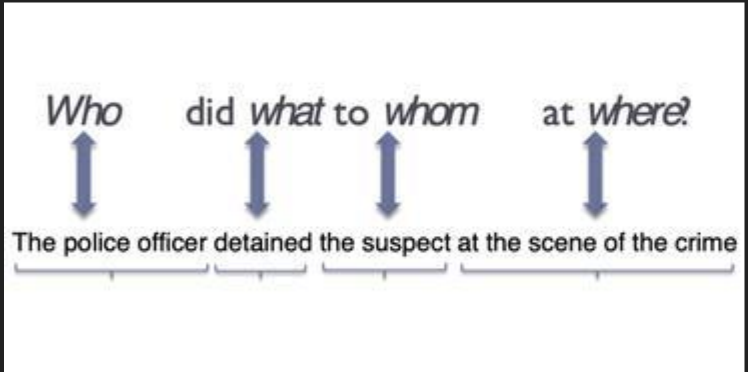
Semantic Role Labelling - Role Labels

In English, some of the important role labels are:

- Agent
- Experiencer
- Theme
- Result
- Location
- Instrument

Semantic Role Labelling - Example

The example shows SRL in English. It shows three labels - Agent, Theme, and Location. In the example, the police officer (**Agent**) is the person detaining (**Predicate**) the suspect (**Theme**) at the scene of the crime (**Location**).



Semantic Role Labelling - Previous Works

- In previous works on SRL in Hindi, many different role labels were used compared to SRL in English. Some are shown below.

Label	**Description**
ARG0	Agent, Experiencer, or doer
ARG1	Patient or Theme
ARG2	Beneficiary
ARG3	Instrument
ARG2-ATR	Attribute or Quality
ARG2-LOC	Physical Location

Dataset - Retrieval and Preprocessing

- We used the Hindi Propbank as the dataset.
- The *.pb* files were processed and Document Objects were extracted.
- Sentence Object is used to parse the sentences into linguistic elements. Each node of the parse tree is processed and we extract features from it.
- All these features are stored in a *.txt* file and then converted into a *.csv* file with “*word, chunk, postposition, head-postag, dependency, is_arg, srl, predicate*” as headers.

Dataset - Retrieval and Preprocessing

- Fasttext embeddings for Hindi language are used for the "head" column words, obtaining 300-dimensional embeddings.
- The embeddings are stored in 300 columns in the CSV file, with each column representing a dimension.
- The final dataset.csv file contains a total of 308 columns, including the embeddings and other features.

Dataset - Retrieval and Preprocessing

This is the structure of the final dataset.csv file.

	dim_1	dim_2	dim_3	dim_4	dim_5	dim_6	dim_7
0	0.020622	-0.043543	0.017362	0.024762	-0.020766	-0.017783	-0.023139
1	-0.014496	-0.008514	0.001838	-0.013815	-0.023862	0.021470	-0.007265
2	0.021526	-0.021832	0.063153	-0.033886	-0.009450	-0.035176	0.080432
3	-0.047248	0.018531	-0.021601	0.054370	-0.024639	0.022130	0.068030
4	0.038407	-0.026316	-0.016674	-0.013441	0.014775	0.005128	0.034026

	dim_8	dim_9	dim_10	...	dim_298	dim_299	dim_300	chunk
0	0.015411	-0.021622	-0.045250	...	-0.002109	-0.018631	-0.002061	NP
1	0.014412	0.024713	0.020350	...	-0.026965	0.007601	0.007057	NP2
2	0.158831	-0.014669	-0.037460	...	0.101427	-0.090182	0.008483	VGNF
3	-0.017688	0.005726	-0.042218	...	0.032187	0.065817	0.067730	NP3
4	0.008274	-0.023595	-0.039830	...	0.047305	-0.050628	-0.014780	NP4

	postposition	head-postag	dependency	is_arg	srl	predicate
0	का	NP2	r6	0.0	22	NaN
1	NaN	VGNF	k2	1.0	2	VGNF
2	हो+एं	NP4	nmod__k1inv	0.0	22	NaN
3	में	VGNF	k7p	1.0	14	VGNF
4	का	VGF	k1	1.0	1	VGF

Argument Classification

It is the task of identifying and classifying the semantic roles of words within a sentence. It involves determining which words in the sentence corresponds to specific semantic roles and assigning the appropriate label of each of them.

In our dataset along with the embeddings of 300 dimensions we also features of "chunk tag , headword of the chunk , POS tag of the headword , Dependency Labels , Postpositions , is_argument" .

For the classification of the model we have tried for the below models

Models

- We trained 4 different models with a train-test split of 85%-15% on the available dataset.
- The models are:
  - Logistic Regression Model
  - Support Vector Machine (SVM) Model
  - Recurrent Neural Network (RNN) Model
  - LSTM (Long Short-Term Memory) Model

Models - Logistic Regression Model

The model was able to achieve 63.4% accuracy.

	A	B	C	D	E
1		precision	recall	f1-score	support
2	1	0.447761194029851	0.158730158730159	0.234375	189
3	2	0.607142857142857	0.299748110831234	0.401349072512648	397
4	3	0	0	0	41
5	4	0.166666666666667	0.0138888888888889	0.0256410256410256	72
6	5	0	0	0	14
7	6	0	0	0	7
8	7	0	0	0	9
9	9	0	0	0	33
10	10	0	0	0	20
11	11	0	0	0	6
12	12	0	0	0	16
13	13	0	0	0	18
14	14	0.333333333333333	0.037593984962406	0.0675675675675676	133
15	15	0.4	0.032258064516129	0.0597014925373134	62
16	16	0	0	0	10
17	19	0	0	0	17
18	21	0.379310344827586	0.159420289855072	0.224489795918367	69
19	22	0.647815912636506	0.938418079096045	0.766497461928934	1770
20	accuracy	0.634408602150538	0.634408602150538	0.634408602150538	0.634408602150538
21	macro avg	0.165668350479822	0.091114309826663	0.0988678564503253	2883
22	weighted avg	0.547902356035521	0.634408602150538	0.551632637249412	2883

Logistic Regression Variations

Using the logistic regression for the simple classification we have used the embeddings and label as SRL, but for the variations we have tried combining the variety of features and evaluated their performance, cause along with embeddings there are other features that help in determining the SRL attribute appropriately

Feature set 1: Embeds + Chunk Tags + Dependency Labels

Feature set 2: Embeds + POS tag + Dependency Labels

Feature set 3: Embeds + Predicate + Dependency Labels

Feature set 4: Embeds + PostPosition + Pos Tag

Results for the Feature Sets

MODEL	ACCURACY
FEATURE SET 1	65 %
FEATURE SET 2	71 %
FEATURE SET 3	85 %
FEATURE SET 4	66 %

Models - SVM Model

The model was able to achieve 63.3% accuracy.

	A	B	C	D	E
1		precision	recall	f1-score	support
2	1	0.403508771929825	0.121693121693122	0.186991869918699	189
3	2	0.689922480620155	0.224181360201511	0.338403041825095	397
4	3	0	0	0	41
5	4	0.333333333333333	0.0138888888888889	0.0266666666666667	72
6	5	0	0	0	14
7	6	0	0	0	7
8	7	0	0	0	9
9	9	0	0	0	33
10	10	0	0	0	20
11	11	0	0	0	6
12	12	0	0	0	16
13	13	0	0	0	18
14	14	0	0	0	133
15	15	0	0	0	62
16	16	0	0	0	10
17	19	0	0	0	17
18	21	0.6	0.0434782608695652	0.0810810810810811	69
19	22	0.636736214605067	0.965536723163842	0.767400089806915	1770
20	accuracy	0.633021158515435	0.633021158515435	0.633021158515435	0.633021158515435
21	macro avg	0.147972266693799	0.0760432419342738	0.0778079305165809	2883
22	weighted avg	0.535062602341973	0.633021158515435	0.532605003320163	2883

Models - RNN Model

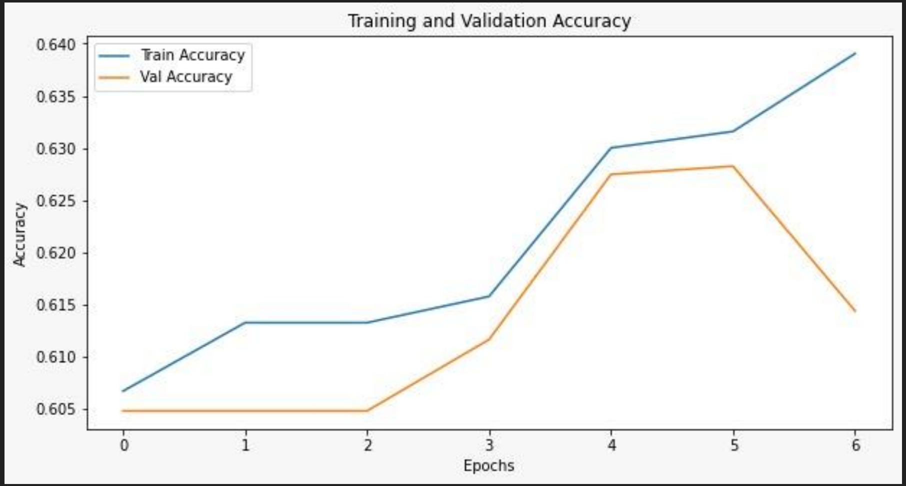
We trained an RNN model with the hyperparameters that follow:

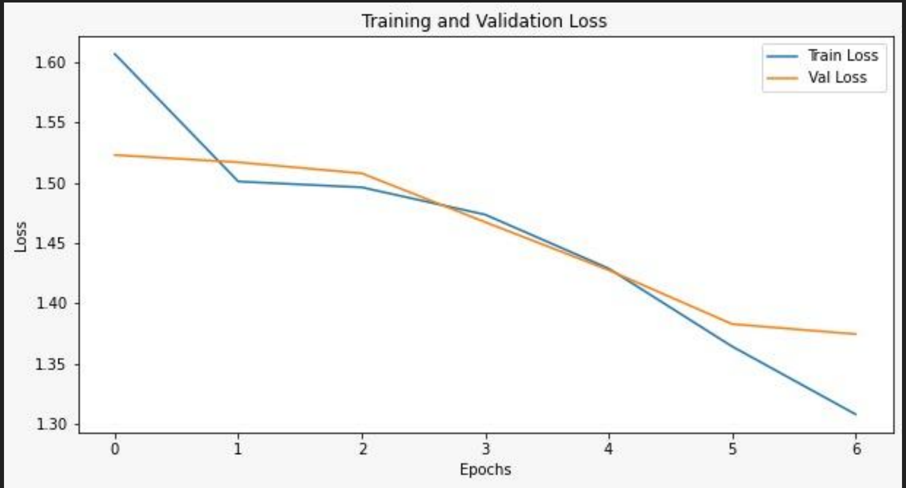
- `EMBEDDING_DIMENSION= 300`
- `HIDDEN_NODES = 200`
- `BATCH_SIZE = 128`
- `Epochs = 6`
- `learning_rate = 0.0005`

LOSS = CROSSENTROPYLOSS

OPTIMIZER = ADAM

Models - RNN Model





Models - RNN Model

The model was able to achieve 62.5% accuracy.

	precision	recall	f1-score	support
1	0.536585365853659	0.530120481927711	0.533333333333333	83
2	0.616666666666667	0.549881656804734	0.600253164556962	169
3	0	0	0	14
4	1	0.0303030303030303	0.0588235294117647	33
5	0	0	0	7
6	0	0	0	2
7	0	0	0	3
8	0	0	0	1
9	0	0	0	10
10	0	0	0	7
11	0	0	0	1
12	0	0	0	7
13	0	0	0	7
14	0.555555555555556	0.317460317460317	0.404040404040404	63
15	0.5	0.115384615384615	0.1875	26
16	0	0	0	7
18	0	0	0	1
19	0	0	0	14
20	0	0	0	1
21	0.655172413793103	0.59375	0.622950819672131	32
22	0.643228602383532	0.730800542740841	0.726506024096386	737
accuracy	0.624489795918367	0.624489795918367	0.624489795918367	0.624489795918367
macro avg	0.0747012237203117	0.0704937993924349	0.0662752514615381	1225
weighted avg	0.481680802846817	0.624489795918367	0.522604170168741	1225

Models - LSTM Model

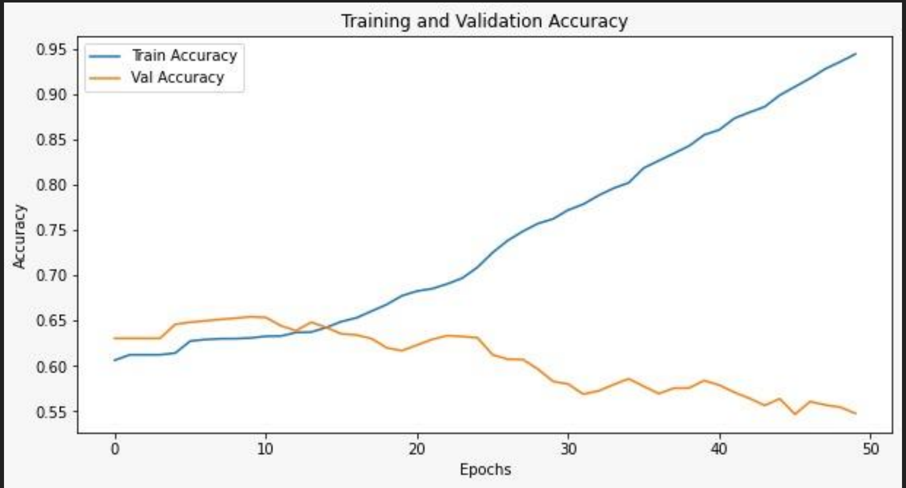
We trained an LSTM model with the hyperparameters that follow:

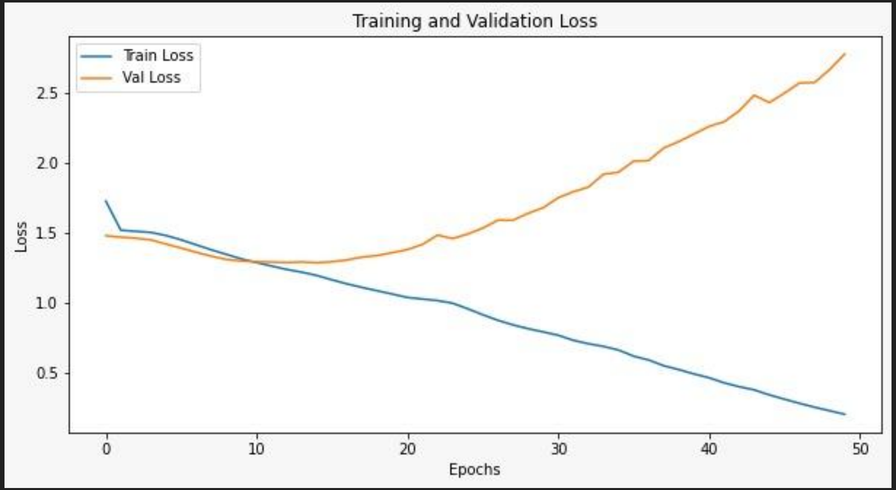
- `EMBEDDING_DIMENSION= 300`
- `HIDDEN_NODES = 200`
- `BATCH_SIZE = 128`
- `Epochs = 50`
- `learning_rate = 0.0005`

LOSS = CROSSENTROPYLOSS

OPTIMIZER = ADAM

Models - LSTM Model





Models - LSTM Model

The model was able to achieve 66.9% accuracy.

	precision	recall	f1-score	support
1	0	0	0	81
2	0.716417910447761	0.266666666666667	0.388663967611336	180
3	0	0	0	16
4	0	0	0	20
5	0	0	0	4
6	0	0	0	4
7	0	0	0	8
9	1	0.0625	0.117647058823529	16
10	0	0	0	7
12	0	0	0	8
13	0	0	0	8
14	0.24	0.46	0.428	50
15	0.466666666666667	0.288888888888889	0.391228070175439	36
16	0	0	0	2
18	0	0	0	2
19	0	0	0	20
21	0	0	0	30
22	0.624352331606218	0.975708502024292	0.761453396524487	741
accuracy	0.669387755102041	0.669387755102041	0.669387755102041	0.669387755102041
macro avg	0.0744872356696655	0.0690208427050532	0.0638954091186568	1225
weighted avg	0.482939021714942	0.629387755102041	0.517711413056886	1225

Conclusion

- Incorporating additional features alongside embeddings significantly improves classification performance.
- SRL label as target variable yields higher accuracies, indicating a strong relationship with dependency tag and predicate.
- Utilized RNN and LSTM architectures for multi-class classification.
- LSTM with bidirectionality outperformed RNN, suggesting its superior ability to capture bidirectional dependencies within input sequences.

Code and References

Code: https://github.com/Hrishikesh0511/Logical-parsers-INLP-Project-SRL-

References: https://www.aclweb.org/anthology/L16-1727.pdf
http://lrec-conf.org/workshops/lrec2018/W29/pdf/28_W29.pdf