Semantic Role Labeling for Indian languages

Thesis submitted in partial fulfillment
of the requirements for the degree of

Masters of Science
in
Computer Science and Engineering by Research

by

Aishwary Gupta
201302216
aishwary.gupta@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
September 2019

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Semantic Role Labeling for Indian languages" by Aishwary Gupta, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Adviser: Dr. Manish Shrivastava

To My Parents, Family and Friends.

Acknowledgments

I would like to thank everyone in my life for the support and inspiration needed for the completion of my research. I would like to express my sincere gratitude to Prof. Manish Shrivastava. I feel very lucky to have him as my advisor. I would like to thank him for his constant guidance, thoughtful insights, patience and for his immense faith in me without which this research wouldn't have been possible. He has a very deep knowledge in linguistics, Natural Language Processing and Machine Learning.
I would like to thank my mentor, Maaz Anwar Nomani for his valuable suggestions and guidance during this dissertation. I would also like to thank Avijit Vajpayee and Syed Sarfaraz Akhtar. They have been great mentors and even greater friends without whose guidance I would not be this knowledgeable in NLP and ML.
I would also like to extend my thanks to my friends Aditya, Akshay, Ankush, Ashutosh, Ayush, Deepanshu, Danish, Gorang and Sahil for providing constant support through the course of thesis as well as keeping me motivated throughout college life.
Last but not the least, I would like to thank my mother, father and brother for their unconditional love, support and advice. I would like to thank them for keeping me highly motivated and giving me freedom and believing in me. You are the reason to make this possible by providing me everything in life and putting so many efforts for me.

Abstract

Semantic role labeling, also known as shallow semantic parsing is a task in Natural Language Processing to determine labels of the words or the phrases (group of words) in a sentence. The labels are of the type agent, patient/receiver, goal, temporal, locative or object. It comes under the domain of Artificial Intelligence for the reason that we want to do this task automatically. Formally, the task can be broken into various steps. The first step is to identify the verbs or the predicates in a sentence. The second step consists of detection of the semantic arguments associated with the corresponding predicates in the sentence and the final step would be to label these arguments pertaining to their respective predicates. For instance, in the sentence, "Ram killed Shyam with a gun", the semantic role labeler should be able to recognize 'killed' (represents the phrase "to kill") as a predicate. Then 'Ram' as representing the killer (agent), 'Shyam' as the recipient/receiver and "a gun" as representing the theme/object.

Such representations are an important step towards figuring the meaning of a sentence. Semantic role labeling (SRL) gives a lower level of abstraction compared to a full syntactic parsing of a sentence. Therefore, it has higher number of classes and groups fewer clauses in each class. For example, "the car belongs to Ram" will need two labels like 'owned' and 'owner' whereas "the car was sold to Ram" will need two different labels altogether such as 'theme/goal' and 'receiver' even though these two clauses would be similar if we use *subject* and *object* functions. Hence, the labels largely depend on the whole clause rather than just the individual phrase or chunk.

In this thesis, we begin with a detailed overview of the literature in the field of semantic role labeling. This includes discussion over the various techniques used to tackle shallow semantic parsing in the past, the development of different datasets built for semantic analysis such as the Propbank and the Framenet and at last we talk about semantic role labeling task for Indian languages. Since most of the Indian Languages are very low-resourced, we have kept our work restricted to only Hindi and Urdu, for which the respective propbanks were good in size. We also give a detailed analysis of the Hindi/Urdu Propbank data sections used by us in our experiments.

Next, we present a statistical semantic role labeler for Hindi which is 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 previous baseline for Hindi, mainly for the classification step. We also report significant improvements for Argument Identification task in Urdu. We propose a new baseline system for Indian languages using 5-fold cross-validation and we capture results excluding the 'null' class and including the 'null' class exclusively. Lastly, we give an error analysis of our model compared with the previous baseline and look at both the improvements and limitations of this model.

Finally, we introduce a supervised deep learning model for the same Indian languages namely, Hindi and Urdu which uses minimal syntax and yet improves over our statistical model 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., both 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 slightly syntax-aware and performs very well compared to the other models. At last, we talk about evaluation metrics and analysis of the three models as well as the statistical model.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Natural language processing (NLP) is a field in the domain of Artificial Intelligence (AI) that deals with programming computers in order to process and analyze large amounts of natural language data which is used by humans to communicate with each other. For easy understanding and analysis, a natural text has to be processed at various stages and each stage can be said to be dependent on the previous stages. To begin with, one will start analysis with simple character representations (Morphology analysis) and word-level analysis (Parts-of-speech). This level of analysis would be very important for subsequent syntactic/semantic analysis resulting to representations such as chunks in sentences, parsed trees etc. Using these further, a higher level analysis could be benefited which may represent full documents (Ex.- summarization, coreference, discourse analysis etc.). Therefore, in a way, each stage is dependent on the correctness of the previous stage. Use of this multi-level information depends on what our end goal is. For example, syntactic parsing will depend on character level (Morph) as well as word level information (Parts-of-speech etc.). In a similar manner, semantic parsing uses the information from syntactic parse tree.

Challenging topics in NLP involve speech recognition, natural language generation and natural language understanding. Semantic role labeling falls under the umbrella of natural language understanding (NLU). NLU is considered as an AI-complete problem which aims to make the machine's reading comprehension (process and understand text like humans) an automated task.

There has been a significant progress in Semantic Role Labeling (SRL), a sub-task of NLU. SRL deals with detecting events in a sentence such as who did what, to whom, where, when etc.

In this thesis, we approach semantic understanding in two different ways. First, we see how the information from syntactic analysis helps to predict the semantic structure of a sentence. Next, we address the same problem by using modern deep learning architectures without relying on any or minimal level of syntactic information. We analyze the two approaches and look at the importance of syntactic information in shaping the semantic structure of a sentence.

1.1 Motivation

In the last two decades, there has been a lot of interest and a great amount of contribution towards semantic analysis of languages. 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 was shown that the knowledge from these semantic frames can be utilized to improve other fields in natural language understanding such as Information Extraction [62, 18, 4], Question Answering [45, 60, 48] and Machine Translation [13, 37], Coreference Resolution [50] and Text Summarization [42].

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. There has been a significant amount of work done for languages like English, Chinese etc. which included efforts in making state-of-the-art role labelers and also included creation of semantically annotated data like Framenet [3] and PropBank [33]. Though much work has been done towards SRL for these resource-rich languages but SRL for Indian Languages (ILs) was pioneered quite recently by Anwar and Sharma [2]. Their system is based on traditional approaches towards SRL as seen in one of the earliest works by Gildea and Jurafsky [22] and Xue and Palmer [70]. They have used a small subset of the Hindi and the Urdu Propbank for training and testing of their model. Thereby, only a little effort can be seen towards semantic understanding in Indian languages.

We saw a need for improvement in this domain and thus we introduce various techniques to address the role labeling problem using - statistical machine learning as well as deep learning neural networks. We also use an additional set of features which makes a significant improvement in the classification of semantic roles for Hindi and we extend the same work for Urdu. Anwar and Sharma [2] made the first attempt at SRL for ILs, there was no prior data to estimate their performance. We compare our models with them and give an analysis on the errors made by their model and how we correct them.

The major objective of SRL is to provide all sorts of semantic 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 (or chunk[*]) level or at a token level. The verb is the predicate and phrases/tokens related to it are its arguments labeled in categories such as agent/doer, patient/receiver, location, temporal etc. A semantic role labeler has to therefore, label the arguments for each predicate of a sentence automatically.

1.2 Syntax

Syntax corresponds to the set of rules of forming a language. It covers the linguistic arrangements of words inside sentences, how we use them in text and their usage in speech. Syntactic structure concerns the right structure of sentences in concurrence with specific standards, procedures and guidelines as set by language experts/grammarians.

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

Syntactic analysis entails various tasks which either have direct applications in the real-world or they act as an intermediate step to one such task. For example, *lemmatization*, which is the task of grouping all inflected forms of a word so they fall under a single lemma and can be analyzed as a single item now. Words like ‘run’, ‘running’, ‘runs’, ‘ran’ will be grouped under the same category which shall be called as the ‘run’ lemma. Lemmatization is better than *stemming* since it also looks at the context. Ex.- let us say ‘meeting’ appears as a noun in some text. A stemmer will convert this to ‘meet’ (to meet) considering it a verb which is clearly wrong. Though, it depends on the end application where these sub-tasks are to be used. In case of information retrieval, a stemmer is always preferred since it is quite faster than lemmatizer and the negative effect on results is not quite significant.

Morphology deals with the analysis of the structure of words in a language. It helps analyzing word level linguistic features like the root form, prefix, suffix, gender, case, person etc.

Parts-Of-Speech (POS) tagging is the procedure to determine the part of speech for each word in a given sentence. The POS tags (noun, verb, adverb, adjective etc.) are assigned depending not only on the word but also the context since words can have multiple parts of speech. Example, the word ‘cook’ can be a noun (“person who prepares food”) and also a verb (“to cook food”). Such ambiguities are more prevalent in languages like English. Hindi and Urdu are morphologically rich languages yet there are few such examples in them. For instance, consider the below example. In 1.1, ‘*aam*’ is used as an adjective while in 1.2, it is used as a noun.

(1.1)	Aisa	hona	***aam***	baat	hai.
	This	happen	***usual***	thing	be
	This is a usual thing.				

(1.2)	Yeh	***aam***	ka	ped	hai.
	This	***Mango***	of	tree	be
	This is a mango tree.				

Parsing has the aim to determine the parse tree (grammatical analysis) of a given sentence conforming to the rules of a formal grammar. Parse tree shows the sentence parsed into constituents which are connected to each other via syntactic relations. Natural texts are difficult to parse because of the complexity and ambiguity in a language. There could be multiple potential parses for a given sentence, each following a different way (grammar rules) for parsing. Hence, there are two primary kinds of parsing followed in general, Constituency parsing and Dependency parsing. Constituency parsing constructs a parse tree based on a Context-Free Grammar (CFG), a Probabilistic CFG to be more precise. Dependency parsing on the other hand uses relationships between words in a given sentence.

There has been a considerable amount of research for parsing of Indian languages and consequently, state-of-the-art parsers have been built for languages like Hindi, Urdu, etc. Hindi/Urdu are parsed using dependency parsing whose grammar is based on Computational Paninian Grammar (CPG) [5]. The

Paninian grammar was created around two thousand years back. According to dependency grammar, words are connected to one another with direct links called dependencies. Usually, the verb is taken as the root unit and all other words/syntactic units in the sentence are connected to it, either directly or indirectly. Dependency grammars are more convenient for free-word order languages like Indian languages. Thus, the Hindi/Urdu Treebank dataset containing thousands of dependency parsed sentences is built using the CPG [11].
In the last decade, dependency-based representations have received a growing amount of attention both by the additions in productivity in investigating the structure of the sentence to expel vagueness and by its potential value in various NLP applications such as semantic role labeling, machine translation, dialogue systems and discourse analysis. We leverage information from syntactic parses, dependency parses in case of Indian languages to construct features for training our semantic role labeler.

1.3 Semantic Analysis

Semantics is term used for study of meaning. In this thesis, we are concerned with 'lexical semantics', which deals with the meaning of words, phrases and interpretation of a text. Semantic analysis is more complex than syntactic analysis of text because it delves deeper into the linguistic aspects of a language. Direct applications of semantic understanding would include real-world problems like Machine translation (Automatic translation of one human language to another), Natural language generation (Generate human readable language from some computational analysis of data), Question-Answering (Answer to a human language question), Information Extraction, sentiment analysis (identify sentiments from public opinion texts), Word Sense Disambiguation (amongst different meanings of a word - like in 'read a book' or 'book a ticket', select the meaning with most sense in a context), etc. Sentence 1.1 and 1.2 can also be looked as an example with different senses of the word '*aam*'.
To make semantic understanding easier, an intermediate task was created called as shallow semantic parsing which is now well known as Semantic role labeling. The idea was to have a lower-level abstraction of events occurring in a sentence. Formally, the task is defined as - Given a sentence, we need to label all the arguments (words or phrases) pertaining to each predicate (verb or sometimes even noun) or falling under the same clause into defined classes which represent agent, receiver, time, location etc. Example of a shallow parsed sentence is shown below.

(1.3)	[Ram]$_{A0}$	[ghar par]$_{AM-LOC}$	[taash]$_{A1}$	[khel]$_{V}$	raha hain
	Ram	home at	cards	playing	be

Ram is playing cards at home.

For simplicity, we have chosen an example with single predicate which is '**kela**' *(to play)*. In this sentence, 'Ram' is labeled as the *agent* because he is causing the event to occur. 'taash' can be easily perceived as the *patient*. "ghar par" shows the locative subjectivity in the sentence. As a result, it is quite evident that we have information such as who is playing what and where. Clearly, this information can aid in tasks such as question answering as we directly answer the questions in the event. Also, it has direct application in information extraction since we detect all relevant data in the event. Similarly, we can also use this information to generate features for tasks such as machine translation and auto-text summarization, thus acting as an intermediate step/sub-task for the final goal just like using the syntactic features.

1.4 Contributions Of This Thesis

The key contributions in this thesis are:

1. A thorough review about the research in this field in the last two decades. We discuss the various approaches that have been used in the past to address SRL. This includes discussion about the whole top to bottom process starting from predicate identification to re-ranking argument classifications. More about this is shown later in this thesis. Then we explore the type of datasets that have been used to aid the development of state-of-the-art role labelers. This includes well known datasets like the FrameNet, which was the first ever data set with semantic frames and annotations. Following this, the PropBank was developed which is the most widely used data till date for SRL and is available in constituency parses as well as dependency parses. Finally, we study and analyze the Hindi/Urdu Treebank and Propbank. We provide comprehensive statistics for the part of Hindi-Urdu Propbank used by us in this work containing details which were not provided in the previous work by Anwar and Sharma [2] on SRL for ILs.

2. We introduce a better statistical supervised semantic role labeler for Indian languages, namely Hindi and Urdu. According to the best of our knowledge, only Anwar and Sharma [2] has worked on SRL for Indian languages prior to us. Our system uses a two-step approach where 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. This approach is similar to Anwar and Sharma's [2] which makes our comparison with them a lot easier since we can directly see the difference in performance as a result of using an additional set of features. We make a significant improvement in the classification of semantic roles for Hindi and extend the same work for Urdu. We also perform 5-fold cross validation of our system and provide the results of only the argument-classes too (excluding NULL class).

3. Finally, we try to apply relatively recent approaches for SRL in ILs 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 with 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, first (a) perform Argument Identification i.e., binary classification and then (b) Argument Classification on the probable arguments passed on by step a. 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 [53]. 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 in earlier systems [55]. This model performs slightly better than the statistical based model discussed above. 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 Anwar and Sharma's work [2] 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 models and baseline. Model C, along with the raw token 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.

1.5 Thesis Organization

This thesis is organized in six chapters. In chapter two, we discuss the previous work relevant to semantic role labeling. We talk about various approaches that have been used to tackle SRL including different techniques that have been used for handling each stage in the whole process.
Chapter three gives a brief outline about the different datasets that have been prevalent in major languages. Then we describe the Hindi and Urdu Propbank project and give an explicit analysis of the sections of dataset used for our experiments.
Chapter four presents our efforts on building a statistical semantic role labeler for Hindi and Urdu which performs better than the previous baseline. We talk about the features used for the problem and show the effects of using each new feature used by us. We perform a 5 fold cross validation to justify the training and testing split of data used. Finally, we analyze the results and the errors in it with respect to the previous baseline.
In Chapter five, we explore deep learning methods for the same task. We present three new models out of which 2 models use syntax while training whilst one model performs SRL without using any syntactic information. Even though deep learning networks require a great amount of data to perform well, all these models perform better than our statistical model on the same set of data. We analyze these three models as well as our statistical model in this chapter.
Finally, in chapter six we conclude our thesis with a brief summary and we look at future work in this domain.

Chapter 2

Related Work

Daniel Gildea and Daniel Jurafsky were the first ones to develop an automatic semantic role labeling system [22]. The creation of the Framenet dataset [3] and the progress in statistical learning helped them come up with their system. The FrameNet project produced the first major computational lexicon that systematically described many predicates and their corresponding roles. They were also the first to introduce the two way approach to SRL, identification of argument constituents and then labeling the identified arguments. Their system used phrase's syntactic category, syntactic label, position of constituent with respect to the predicate, voice type and head word as features. A list of features which have been used by various systems till now for argument identification and argument classification is shown in Table 2.1.

Their system did not make use of any definitive machine learning algorithm as we know them today. They rather used probability and linear interpolation to create their own predictive mathematical function. No later, the PropBank [33] corpus was announced for English which added manually created semantic role annotations to the Penn TreeBank corpus of Wall Street Journal texts [39]. Since then, fast growth has been made in this field and many automatic semantic role labeling systems have used PropBank as a training dataset to learn how to annotate new sentences automatically. This acted as a foundation stone in the field of shallow semantic parsing/semantic role labelling.

This growth can be attributed to better modeling techniques using sophisticated machine learning algorithms, more relevant feature engineering and most importantly, cleaner annotation. Following the work of Gildea and Jurafsky [22], Gildea and Palmer [23] built the first system trained on the Propbank. They reported results on three evaluations - The first two report results on gold standard parses, one assuming that the constituents which are semantic arguments are given and another assuming that first the arguments have to be identified too. The third evaluation is done to depict automatic SRL and hence it is done on a system built using automatic parses which has to first parse sentences and then do role labeling. Subsequently, improvements in this field kept happening by exploring deep linguistic features [17] and the use of machine learning with algorithms like SVM, maximum entropy classifier etc.

Surdeanu et al. [62] used a C5 decision tree classifier and found new features like *content word* (word with most information in a constituent), its POS tag, named entity etc.

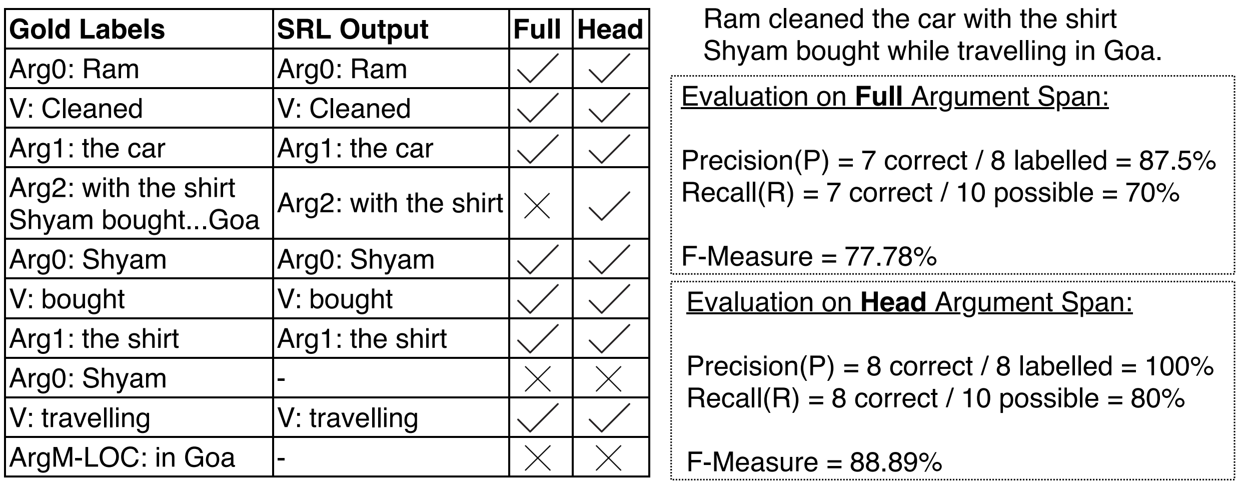
Pradhan et al. [52] used Support vector machine (SVM) to address the problem along with introducing new features like POS tag of head word, named entities, using only partial path, head word of prepositional phrase etc. Extensive feature engineering like theirs lead to better results than the previous baseline.
Xue and Palmer [70] made a crucial contribution to filter out constituents which are definitely not going to be arguments with respect to a predicate. Most works have been using their filtration step as a pre processing. For Hindi/Urdu, we also make such an attempt shown in chapter 4. They used a maximum entropy classifier (Maxent classier) with simple features going easy on feature extraction.

Commonly Used Features	
• Predicate Lemma	• Predicate's children POS
• Predicate's POS	• Predicate's parent POS
• Argument Word/Chunk POS	• Argument's children POS
• Phrase Head word	• Argument's parent POS
• Head word POS	• Voice
• Words in the chunk	• Position w.r.t. predicate.
• Context from left and right chunks	• Named Entity Category
• Dependency label	• Distance from predicate
• Dependency path from predicate to argument	• Word Sense
• POS tags path sequence from predicate to argument	
• Predicate's and argument's children words and parent words	

Table 2.1 Features commonly for training identification and classification of arguments for semantic role labeling [22, 55, 66, 70]. Note that dependency label and dependency path are used as features specific to dependency based SRL. Span based SRL uses all features except these two.

2.1 Shared tasks for SRL

During this era, Semantic role labeling kept gaining popularity and as an out-turn, a shared task (CoNLL 2004/05 shared task [15, 16]) was announced where we saw one of the best systems/approaches to the problem of constituent/span based role labeling. The task was now formally defined with official release of training, development and test sections. The evaluation was more legitimate now since the final score was now calculated on the basis of number of correct roles assigned in the test data. The systems can be trained on the training data and the hyper-parameters and optimizations can be done on the validation set so that the test data remains unseen. Correct roles meant the span boundary as well as the label both had to be correct. This is done by comparing the output labels for a sentence given by the system with the labels of this sentence in the gold parse. The example below in Figure 2.1 explains a dummy evaluation for SRL.
Later on, another shared task was announced for dependency based SRL where we only label the head words as arguments [63].



Gold Labels	SRL Output	Full	Head
Arg0: Ram	Arg0: Ram	✓	✓
V: Cleaned	V: Cleaned	✓	✓
Arg1: the car	Arg1: the car	✓	✓
Arg2: with the shirt Shyam bought...Goa	Arg2: with the shirt	✗	✓
Arg0: Shyam	Arg0: Shyam	✓	✓
V: bought	V: bought	✓	✓
Arg1: the shirt	Arg1: the shirt	✓	✓
Arg0: Shyam	-	✗	✗
V: travelling	V: travelling	✓	✓
ArgM-LOC: in Goa	-	✗	✗

Ram cleaned the car with the shirt
Shyam bought while travelling in Goa.

Evaluation on **Full** Argument Span:

Precision(P) = 7 correct / 8 labelled = 87.5%
Recall(R) = 7 correct / 10 possible = 70%

F-Measure = 77.78%

Evaluation on **Head** Argument Span:

Precision(P) = 8 correct / 8 labelled = 100%
Recall(R) = 8 correct / 10 possible = 80%

F-Measure = 88.89%

Figure 2.1 Evaluation methods for Semantic role labeling

2.2 Machine Learning methods

People have applied different methods to train their models which includes maximum entropy classifier [67, 72, 10], SVM using polynomial kernels [51] or Gaussian kernels [47]. Punyakanok et al. [34] used the SNoW (Sparse Network of Winnows) learning architecture, which is a sparse network of linear separators. Decision Trees like C4.5 [49], decision tree ensembles [40, 65] were used with AdaBoost optimization.

From the Machine Learning point of view, model combination/ensembling has been shown to improve results. Output combinations can be developed by either varying inputs to your model (by providing different feature sets to the different unit models), having different learning methods or creating n-best solutions lists. These have been done in the past using simple heuristics like voting model, stacking models for classification etc. Pradhan et al. [51] followed a stacking approach by learning a chunk-based SRL system including as features the outputs of two syntax-based systems.

Surdeanu et al. [64], Punyakanok et al. [53] and Toutanova et al. [68] have also created ensemble variants of their systems by using multiple systems or multiple parse trees with same system. Fitzgerald et al. [21] used a product-of-experts model [29] which sums factors of the potential functions from different trained neural networks produced by random initializations of parameters (of the same model) over different runs.

Most systems approach SRL by dividing it into two steps - first to identify arguments with a binary classifier (null vs non-null) and second step is to classify the arguments which passed through step one. Some systems also treated this as a usual multi-class classification problem where single step classification is done. Some people have also performed the task as a sequence tagging problem using IOB (BIO) tagging scheme. We will talk about this later in Section 2.5 of this chapter.

Moschitti et al. [44] presented an exceptional strategy which divides the two steps, identification and labeling of arguments into 4 steps. After performing argument identification, they applied heuristics to check compatibility of identified arguments among one another and before argument classification they applied a pre-classification step to classify between core labels and adjunct labels. Another special formulation was done by Bharati et al. [10], who changed the identification step as classification into three classes - mandatory, optional and null.
Post classification, many systems started using scoring functions on top candidate arguments to make the results global which further optimizes the results. This is called the Inferencing stage and discussed in the next section.

2.3 Globalization and Inferencing methods

Global constraint satisfaction framework used to re-rank alternative outputs represent a very interesting alternative [16]. Gildea and Jurafsky [22], have introduced a prior in their probabilistic model to integrate structural dependencies. They included a pre-calculated probability distribution over the multi-set (S) of label groups that belongs to the predicate being considered. Therefore, if some unlikely/unusual roles are assigned by the local model, the results would get adjusted according to distribution of role groups in S. Probability distribution over S is estimated using interpolation of a relative frequency and a back-off distribution. Bernoulli's Naive Bayes model is assumed to find the back-off distribution and hence it assumes each argument label is present or absent independently of the other labels. So now, re-ranking of top results from the local model would give the most likely assignment but this approach does not model the dependencies amongst the arguments, structural or linguistic constraints as shown in Table 2.2 below.

Constraints in role labeling
• Arguments cannot overlap with the predicate
• Arguments cannot exclusively overlap with the clauses
• If a predicate is outside a clause, its arguments cannot be embedded in that clause
• No overlapping in the arguments
• No duplicate argument classes for core arguments/numbered arguments, such as A0-A5
• Reference arguments can occur only if referenced argument is present
• Continuation arguments can occur only if base argument is present
• Arguments should also conform to frame file rules of the predicate

Table 2.2 Widely accepted structural and linguistic constraints in SRL.

Pradhan et al. [51] have also tried to include inter-dependencies by including features such as dynamic context (using labels of left *k* nodes as feature). They also trained a trigram language model on argument

sequences and combine this with the probabilities they get after classification. Then they used *Viterbi* search over a generated argument lattice to find the maximum likelihood path.
Toutonova et al. [67] addressed this problem by building a joint re-ranking model on top of a local model. The re-ranking model takes as input the top-*k* assignments of labels from the local model and uses it with features like actual label sequence of the concerning predicate and even all features from the local model. The top-*k* assignments are obtained by a dynamic program which computes an argument lattice similar to Viterbi decoding. The final step does a maximum likelihood computation by combining (taking product) the re-ranked model probabilities and local model's probabilities. Bjorkelund et al. [12] used a similar approach to build a global re-ranker.
Global inference with constraint satisfaction has been done performed using techniques like *Integer Linear Programming* (ILP) [54]. The inference considers all candidate arguments can be labeled for each constituent. The global inference not only gives the best global solution for a proposition but also takes into account linguistic and structural constraints which makes the inferences more reasonable.
Early models gave the results after argument classification step. The problem here is that label for arguments are decided independently without taking into consideration the global information. When accounting that information structural constraints such as numbered arguments (Arg0, Arg1 etc) can be assigned to single argument for a predicate and argument spans should not overlap etc. have to be applied via some formulation on the scores output by the classification stage.
Tackstorm et al. [66] have presented a *Dynamic Program* (DP) formulation to enforce the constraints showed by Punyakanok et al. [53]. Their system gives identical results as an ILP solver, performing much faster than it. Their model trains a global log-linear model which enforces the structural constraints during training only.

2.4 Dependency based SRL

Semantic role labeling on a dependency parse was introduced by Hacioglu [25]. For the English propbank, the task is to label only the syntactic heads and not the full constituent. In case of Hindi/Urdu, dependency parse is slightly different from what we see in English as shown below in Figure 2.2.
Official task for dependency based SRL was announced [63, 26] which not only focused on English but other languages like Spanish, Catalan, Chinese etc. For this task, the data was same as used in the 2005 Shared task [16] release but all the sentences were converted from constituency parse to dependency parse. The shared task also introduced nominal predicates to Propbank from the Penn Treebank data.

The idea in dependency parsing is to make a tree where each node spans only a word itself connected in the upper tree from another single word and in the subtree it has all the words it anchors. Thus, intermediate representations covering only some dependents do not directly correspond to anything in a dependency tree. Due to the simplicity, quicker computation and closeness to semantic representation of sentences, there has been an increase in the use of dependency parses for NLP applications [41, 46].

Though this task is different from ours (span based role labeling) but the Hindi/Urdu dataset share a very important similarity with the annotations introduced in the 2008 shared task [63]. Both the English dataset and Indian languages dataset have a dependency based syntactic structure. But the difference lies in the fact that in the Hindi/Urdu propbank the sentences given are parsed and also role labeled at a phrase/chunk level while in the 2008 shared task data for English and 2009 shared task for other languages as well, the sentences are fully parsed till node level where the leaf node in the dependency tree is always a token and hence only the head words get labeled with semantic roles. The example below shows the differences in these datasets.

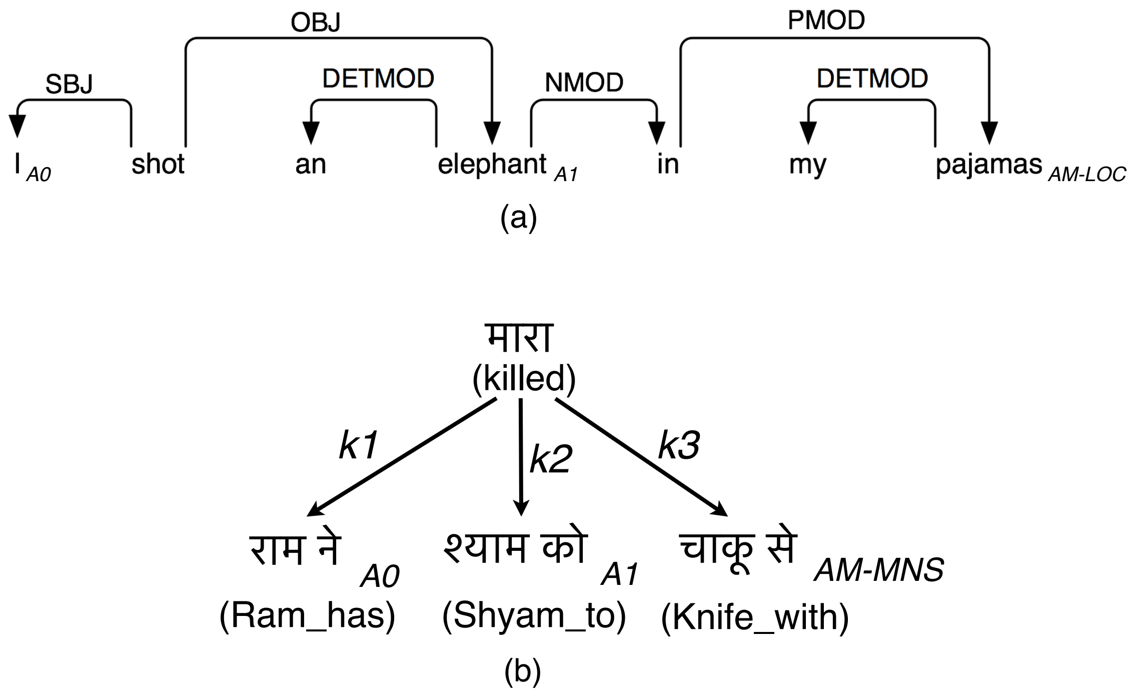


Figure 2.2 Dependency parsed sentence in (a) English and (b) Hindi.

Similar approaches were seen to tackle semantic role labeling for this dataset as well. And apparently, syntactic features such as node relations, path etc. were similarly extracted from the dependency parses in place of the syntactic parses. Likewise, features like argument word, left word, right word and their POS tags could be used.

2.5 Neural Networks and Deep Learning

Collobert and Weston [19] was the first one to apply Multi Layered Perceptron (MLP) to SRL. They built a neural network model for multitask learning which learns semantic roles for words/chunks in a sentence without using a syntactic parser and chunker. They built a convolutional neural network (CNN) architecture which along with SRL, jointly models NLP tasks like POS tagging, Chunking, Named-entity recognition, Language modeling etc.
Low-dimensional embedding representations have been shown to be successful in overcoming sparsity and representing label similarity across a wide range of tasks [20, 61, 28, 35]. Roth and Woodsend [56] augmented the feature space with word representations for head word of arguments and the predicate being considered. They learn representation of the full argument span by combining all words in it and also the representation of path between argument head and predicate by encoding all words between them.
Fitzgerald et al. [21] jointly embedded candidate arguments and semantic roles for a given predicate/frame in a shared vector space. The embeddings are created using a neural network whose final output combines the embedding of candidates and the role embeddings.
In the last few years, significant work has been done towards fully syntax-agnostic approaches using deep neural networks. Collobert et al. [20] 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 [74] whose system differs from Collobert et al.'s [20] 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. [27] 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 sentence's 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. [38] did a similar approach for dependency based SRL. Almost all of the above span-based systems have used an '**IOB**' labeling on the data. IOB stands for **I**nside, **O**utside and **B**eginning of an argument span. This is shown later via Figure 5.3 in chapter 5. Before these systems became popular, neural networks were used in syntax-aware systems also [21, 66, 55].

2.6 Indian SRL

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 done by Anwar and Sharma [2]. Since a single system has been made for semantic role labeling for Indian Languages, we take it as the best model available and compare our models with them. They use simple features like dependency

labels, syntactic categories, head word of the chunk, head word's POS tag, Named entities etc. In chapter 4, we propose a model which improves the labeler by introducing features like word embeddings to address the data sparsity issue, path from chunk to predicate and post-positionals of the chunk.
In chapter 5, 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. [69] has shown that the predicate-argument structure is closely related to dependency relations. The same was proven in the system by Anwar and Sharma [2] 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 also build a fully syntax-agnostic model to eliminate our reliance on syntax.

2.7 Conclusion

Semantic role labeling has been a highly popular task in semantic analysis and has been investigated using various different methods and for different languages over the years.
In this chapter, we had a look at the basic statistical models that have been used to solve SRL which led to discovering many important features and factors that play a good role in model training. Then we discussed how inferencing of results could be constrained with the linguistic and structural properties of a language. We also discussed about various types of labeling and how is it done in the context of Indian languages.
Then, we talked about how deep learning has been a breakthrough in the field of semantic analysis. At last, we look at the prior work that has been done for Indian languages.

Chapter 3

Datasets for Semantic Analysis

3.1 Foundation

Levin [36] made the first effort in linking semantic roles and syntactic structures. They defined verb classes also known as *'Levin classes'* which groups verbs based on their ability to occur in a syntactic frame. Therefore, the verb-syntactic frame pair fall in a Levin class which shares semantic properties amongst all verbs belonging in this class. For example, break/shatter/smash reflect similar semantic components. *VerbNet* [58] added an additional layer over Levin's classes which abstracts the representation of syntactic frame of each class by making relations between syntactic positions of chunks and semantic roles they can have.
Subsequently, the *FrameNet* project [3] proposed roles specific to certain semantic structures which they called as semantic frames. The semantic frames describe an event, the participants, relations and objects in it. Framenet roles/Frame elements are defined for each semantic frame or Frameset. Similar verbs are grouped into the same semantic frame based on their semantic similarity. For example, the semantic frame 'Commerce' has the frame elements buyer, seller, goods, money etc. and hence verbs like 'to buy', 'to sell' etc. belong to this frame. It is ensured that this grouping is neither too generic nor too specific. The Framenet was proposed as a domain independent as well as a language independent resource. FrameNet annotation includes nouns and adjectives also. A predicate with multiple word senses generally belongs to different frame for each sense. Gildea and Jurafsky [22] were the first to use FrameNet for SRL.

3.2 The PropBank

Propbank is the most widely used corpus for SRL that is annotated with predicates (verbal/nominal propositions) and their arguments. PropBank was first proposed by Kingsbury and Palmer [33] for English language. Since then, a great amount of work has been done for other languages too (Chinese, Spanish, Catalan etc). We use the Hindi-Urdu Propbank for our experiments. The PropBank and the FrameNet both have the aim to annotate predicates with their semantic roles yet they are quite different.

PropBank annotations are closer to syntax because they are done above a treebank while FrameNet annotations are sometimes more semantically driven, done independent of parse trees. For instance, the English propbank is annotated on top of the Penn Treebank [39] and arguments are the nodes/spans in the actual syntactic trees. Whilst in FrameNet, the annotations are marked with start and end of a frame-element/role independently of any parse tree. The major difference is that the PropBank defines roles for each verb independently i.e., on a verb-by-verb basis, though the roles are kept somewhat with a similar meaning across all the predicates as shown in example below. Sometimes even a different annotation is defined for different senses of the same verb. The arguments are defined as core arguments, adjuncts and modifiers. The core arguments are the numbered arguments which are defined for each predicate distinctively (verb-specific roles like agent, patient etc.). The adjuncts and modifier roles are general roles not dependent on the predicate like temporal, locative roles etc. More detail about roles and their meanings is given in the next section and Table 3.2.
The sentences below can be seen to get an idea about the annotation schemes for FrameNet and PropBank. According to FrameNet, both sentences describe the same event where buyer, seller and goods remain the same. Whilst in case of PropBank, *Arg0* and *Arg2* roles are assigned differently depending on the predicate (verb).

FrameNet Annotation:

(3.1) [Ram]$_{buyer}$ bought [a bike]$_{goods}$ [from Shyam]$_{seller}$ [for 20000 rupees.]$_{payment}$

(3.2) [Shyam]$_{seller}$ sold [a bike]$_{goods}$ [to Ram]$_{buyer}$ [for 20000 rupees.]$_{payment}$

PropBank Annotation:

(3.3) [Ram]$_{Arg0}$ bought [a bike]$_{Arg1}$ [from Shyam]$_{Arg2}$ [for 20000 rupees.]$_{Arg3}$

(3.4) [Shyam]$_{Arg0}$ sold [a bike]$_{Arg1}$ [to Ram]$_{Arg2}$ [for 20000 rupees.]$_{Arg3}$

Earlier annotations in Propbank were done on constituency parses. In, 2008's shared task [63], an effort was made to convert the dataset to dependency trees and thus annotating the syntactic heads with roles. It also added nominal predicates (nouns) to the propbank.

3.3 Hindi and Urdu PropBank

The Hindi PropBank and the Urdu PropBank were proposed as a part of a multi-layered and multi-dimensional resource which includes both dependency annotations as well as semantic annotations on top of the dependency parses [11]. Figure 3.1 shows a Hindi sentence represented as (a) constituency parse and (b) dependency parse. Phrase structure trees/constituency parses can be generated from this corpus using an automated conversion explained in Bhatt et al. [11]. Annotations for the Urdu PropBank were done by Anwar et al. [1]. Unlike PropBanks in most 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 Hindi/Urdu PropBank corpus. The Hindi Dependency Treebank is based on Panini's Sanskrit grammar (also known as CPG: *Computational Paninian Grammar*; [5, 8]). Urdu Dependency Treebank is also based on the CPG [1]. The dependency trees in Hindi-Urdu corpus have dependency relations between chunks/phrases rather than single words as in English dependency trees. Thus, each node in the tree represents a span of words which are inter-related to other nodes spanning a series of words in the tree. Bharati et al. [7] defined a Chunk as - "A minimal (non recursive) phrase(partial structure) consisting of correlated, inseparable words/entities, such that the intra-chunk dependencies are not distorted" . Although, Intra-chunk dependencies can also be constructed automatically using rule based algorithms [31].

Each verb's syntactic dependents are annotated as their semantic arguments at the chunk level itself.

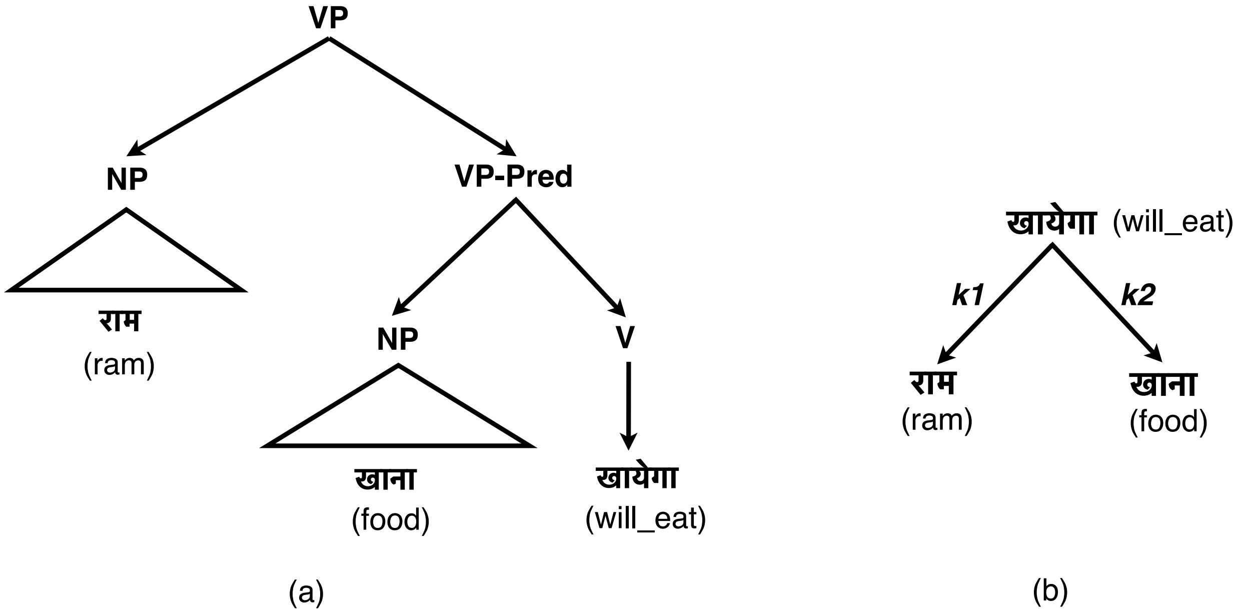


Figure 3.1 Parsing of a sentence in Hindi using (a) Constituency Parsing (b) Dependency Parsing

PropBanks of both the languages include dependency relations at the chunk level, morphological information for each word, chunk boundaries, 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 [9]. 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 [69] 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 # $\in \{0,1,2,3\}$. For example, the verb 'bawA' (to tell), has three 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 is - 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 to avoid it from being semantically overloaded [71]. 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. "ARGC and ARGA mark the arguments of morphological causatives in Hindi, which is different from the ARG0 notion of causer." [69]

Label	Description
ARG0	Agent, causer, experiencer
ARG1	Patient, theme, undergoer
ARG2	Beneficiary
ARG3	Instrument
ARG2-ATR	Attribute
ARG2-GOL	Goal
ARG2-LOC	Location
ARG2-SOU	Source
ARGC	Causer
ARGA	Secondary causer
ARGM-VLV	Verb-verb construction
ARGM-PRX	Noun-verb construction
ARGM-ADV	Adverb
ARGM-CAU	Cause
ARGM-DIR	Direction
ARGM-DIS	Discourse
ARGM-EXT	Extent
ARGM-LOC	Location
ARGM-MNR	Manner
ARGM-MNS	Means
ARGM-MOD	Modal
ARGM-NEG	Negation
ARGM-PRP	Purpose
ARGM-TMP	Temporal

Table 3.1 Hindi/Urdu Propbank labels and their descriptions

3.3.1 Dataset

As reported earlier, for Hindi PropBank, Anwar and Sharma [2] took only a small section for both Hindi and Urdu. We have used exactly same data for Phase 1 of our experiments in chapter 4. For both the languages, we use the same train sections and test sections as in previous work.

	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 3.2 Hindi and Urdu Propbank data statistics.

The dependency relation labels are based on the notion of ‘karaka’, defined as “he role played by a participant in an action” [69]. ‘Karakas’ can also be understood as the syntactico-semantic relations between constituents of a sentence. These syntactic labels are of three types: dependency relation labels,

modifiers and non-dependency labels [11]. ARGC and ARGA are morphological causatives introduced for Hindi. ARGM-VLV and ARGM-PRX (ARGument-PRedicating eXpresstion) are complex predicates used for annotating light verbs in PropBank [32].
Vaidya et al. [69] conducted experiments on Hindi Treebank to automate roles annotation throughout the dataset. Since the treebank which is already labeled with propbank labels is quite small, they did not use a separate set for evaluation. For the same reasons we also do not use a validation set for our experiments in chapter 4 and 5. They defined rules which can be used for automated mapping of syntactic chunks with semantic labels while achieving over 90% accuracy on half the data.

In chapter 4, we 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 by Anwar and Sharma [2]. 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 3.2. The exact data file names used for training and testing of Hindi are listed in the link provided in section 3.4 of this paper. For Urdu, the dataset used was created in an annotation effort carried out in another work by Anwar et al. [1]. In Table 3.2, 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.

3.4 Conclusion

In this chapter, we present details about the data section used by us for our model training and testing purposes.
We also discussed the differences in Framenet style and Propbank style annotations for English and how annotations are done for Hindi/Urdu dependency treebank.
The names of the files in training set and testing set of Hindi are provided in a github repository at the following link. `https://github.com/ashg1910/indian_srl`

Chapter 4

Statistical Semantic Role Labeler for Indian Languages

4.1 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 for these languages was built by Anwar and Sharma [2]. 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 we extend the same work for Urdu.

We did an analysis of their model (shown later in this chapter) which demonstrates the need for improvements. Many of the chunks which are actually arguments are labeled as non-arguments or NULL. Even the core arguments like ARG0 (agent) and ARG1 (patient) are having a high confusion. Other limitations are that they have missed out on using well known features like 'path' and 'post-positionals' which give an improvement in results as seen later in this chapter.

Most of the previous works [51, 54, 34, 2] 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 4.2.1.
Section 4.2 shows our detailed approach and the system architecture. It also talks about the classifier we have used which helps us perform the identification and classification tasks better than the previous baseline by Anwar and Sharma [2]. In Section 4.3, we talk about the current best system as the baseline and then talk about the new features we have proposed. In section 4.4, 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. In section 4.5, We do an error analysis of the previous model as well as our model and try to capture how and where our model performs better than Anwar and Sharma's [2] model.

4.2 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 [2] 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.

4.2.1 Selecting Approach

As shown in the previous work on Indian Languages by Anwar and Sharma [2], 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 [70], 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 Classification step because we are using SVM for our experiments where the training time increases exponentially with number of samples [52]. Hence, we follow the 2 step approach, i.e., identifying the null labels and then classifying the rest. Therefore, we train a binary classifier to label each chunk as a semantic argument or not and remove the ones which are labeled as non-arguments before classification step. For Hindi, this reduces the training data by 51% and for Urdu, there is a large reduction by 81%. Some of the NULL arguments also go to the classification/labeling step (10% for Hindi and less than 1% for Urdu). Similarly, some of the non-NULL arguments are filtered out in the identification step. Finally, we train a multi-class classifier to label the chunks in all classes including the NULL class.

4.2.2 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). This removes majority of NULL arguments from our training data. During the training phase of our model, we train this classifier on entire training set and then apply classification on the training data itself.

Step 2: We run a multi-category classifier on the remaining training set to classify the constituents that are labeled as arguments into one of the classes plus NULL class as few null arguments would have passed step 1.
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 no over-fitting. We also tried our hands using a simple 2 layered neural network having the 1st layer equal to the number of features (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.

4.3 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.

4.3.1 Baseline Features

We take features from Anwar and Sharma’s [2] model 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 features 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 with 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 [2] for Hindi and Urdu. Also, as an inspiration from "Analysis of the Hindi Proposition Bank using Dependency Structure" [69], we incorporate this feature because of the mappings in their work 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.

4.3.2 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:

4.3.2.1 Features for 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 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 [43] 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.

4.3.2.2 Features for 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 suffices in Indian languages. In case of Hindi which uses post-positionals (case markers ‘***ne***’(has), ’***mein***’(in), ’***se***’(from), ’***ko***’(to), ’***ki***’(of), ’***par***’(on)) instead of prepositions as in English, the post-position similarly provides a good discrimination in selecting the semantic labels. The case markers are not helpful in identification of arguments but they provide good information about the token before it and the phrase in which they occur. For instance, the occurrence of ‘ne’ implies the presence of agent/ARG0, ‘ko’ occurs after receiver/ARG1 and ‘mein’ tends to occur in location related context/ARGM-LOC.

Other than these we also tried using *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 [5]. These are namely NULL__NP, NULL__VGF, NULL__CCP. We also appended their corresponding non-null categories. 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.

4.4 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 [2], also shown in Table 3.2. 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 4.1 and table 4.2 respec-

tively. 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
Anwar and Sharma [2]	88	87	87
This work	**91.41**	**90.49**	**90.94**

Table 4.1 Argument-Identification results for Hindi

System	Precision	Recall	F-Score
Anwar and Sharma [2]	78	79	78
This work	**92.05**	**91.49**	**91.76**

Table 4.2 Argument-Identification results for Urdu

Argument Classification. We begin by building the results on baseline features in Hindi shown in table 4.3. 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.

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	60.41	59.93	60.16
+Voice	55.32	49.47	52.23

Table 4.3 Argument-Classification Feature-Wise results for Hindi

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) [43] approach. To train that model on our language, for both Hindi and Urdu we used raw sentences from their corresponding Dependency Tree-Banks. For both languages, we used around 200,000 tokens to train the Word2Vec model and used that model for our feature conversion. Table 4.4 and Table 4.5 shows the comparison of Argument-Classification (including NULL class) for Hindi and Urdu respectively.

System	Precision	Recall	F-Score
Anwar and Sharma [2]	58	42	49
This work	**65.01**	**66.62**	**65.80**

Table 4.4 Argument-Classification results for Hindi

System	Precision	Recall	F-Score
Anwar and Sharma [2]	**87**	85	86
This work	86.72	**86.37**	**86.54**

Table 4.5 Argument-Classification results for Urdu

Phase 2- In phase 1, the data was split in train and test as done in previous work which is a 83.33% train to 16.67% test data for Hindi; and 81.25% train to 18.75% test data for Urdu. 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 4.6 and table 4.7 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 4.6 5-fold Cross-Validation for Hindi.
(* These results exclude NULL class’s contribution in the final score.)

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 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.

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 4.7 5-fold Cross-Validation for Urdu
(* These results exclude NULL class's contribution in the final score.)

Language	Model	Precision	Recall	F1-score
Hindi	Anwar and Sharma [2]	33.98	43.11	38.01
	Our Model	**42.52**	**49.66**	**45.81**
Urdu	Anwar and Sharma [2]	86.12	65.44	74.37
	Our Model	**86.41**	**67.16**	**75.58**

Table 4.8 Combined result for identification and classification of arguments.

4.5 Error Analysis and Discussion

We can see from tables 4.1 and 4.2 and tables 4.4 and 4.5 that our model gives better results for argument identification and argument classification respectively. In this section, we try to analyze our model to reason out the improvements with respect to Anwar and Sharma's [2] system and we also look at the errors which our model is not able to cater to. We did the error analysis for Hindi language only. Tables 4.8 and 4.9 are the confusion matrix for the previous baseline system and our system respectively.

	NULL	**A0**	**A1**	**A2**	**A2-ATR**	**A2-GOL**	**A2-LOC**	**A2-SOU**	**AM-ADV**	**AM-DIR**	**AM-LOC**	**AM-MNR**	**AM-MNS**
NULL	-	187	470	27	46	0	0	0	0	0	231	15	0
A0	88	141	85	5	0	0	0	0	0	0	1	0	0
A1	261	13	273	2	7	0	0	0	0	0	3	0	0
A2	19	0	3	10	0	0	0	0	0	0	1	0	0
A2-ATR	24	0	12	0	40	0	0	0	0	0	0	1	0
A2-GOL	9	0	2	0	0	0	0	0	0	0	2	0	0
A2-LOC	4	0	0	0	0	0	0	0	0	0	7	0	0
A2-SOU	11	0	0	1	0	0	0	0	0	0	2	0	0
AM-ADV	23	0	1	0	0	0	0	0	0	0	8	0	0
AM-DIR	8	0	0	0	0	0	0	0	0	0	0	0	0
AM-LOC	77	0	1	0	0	0	0	0	0	0	131	0	0
AM-MNR	73	0	3	1	4	0	0	0	0	0	13	10	0
AM-MNS	13	0	0	0	0	0	0	0	0	0	0	0	0

Table 4.9 Confusion Matrix for Hindi test data by Anwar and Sharma [2]

Few observations that we can make from the confusion matrix above are: (a) There are a high number of chunks which actually are non-arguments but are getting assigned with various roles. This probably reflects that their identification step is not well trained which can also be seen in Tables 4.1 and 4.2.

We can also look at other rows and see that even some labeled chunks are missed out and given the NULL class. (b) There is a significant confusion in ‘ARG0’ and ‘ARG1’. Almost half of the miss classified ARG0’s are assigned as ARG1. This might due to the fact that the only feature in their system that can differentiate between core argument classes is the dependency relation. All the other features like predicate, head word and chunk type don’t really say much because there are many instances in the data where a chunk is labeled as ARG0 with respect to one predicate but ARG1 with respect to the other predicate or vice-versa, of the same sentence. These features remain the same independent of the predicate being considered. (c) Other label pairs like ‘ARG2-ATR’ getting labeled as ‘ARG1’, ‘ARGM-MNR’ labeled as ‘ARGM-LOC’ and ‘ARG2-LOC’ also labeled as ‘ARGM-LOC’.
Our target is to remove such errors or alteast minimize them as far as possible. The major cause behind these errors is the choice of features which are not able to learn the classification boundaries really well. It is important to use features like path and word embeddings as they provide information about the context as well. Also, the predicate’s morphology gives an idea about the tense and the speech in the sentence. Anwar and Sharma [2] did not experiment with vibhakti/post-positionals as features but they commented that it might bring in more confusion in the argument classification step. Surprisingly, we have found from our experiments that it is a very discriminative feature. (Table 4.3)

Let us look at few examples from the Hindi test data and compare the **SRL labels using Anwar and Sharma’s [2] model** with the gold labels. (Shown below)

(1) *[ek aur kaidi ne] [jail parisher ke shauchalay mei] fansi [lagakar] [atmahatya] (kar)$_V$ li thi*
[Another prisoner has] [Jailer of Toilet in] hanging [suicide] [committing] (did)
SRL: **A1** **AM-LOC** **AM-MNR** **A1**
Gold: **A0** **AM-LOC** **AM-MNR** **A1**
(Another prisoner committed suicide by hanging himself in the Jailer’s toilet room)

(2) *jabki [rishi vaigyanik] (mante)$_V$ hai ki [barish se] faslon ko nuksaan pahuncha hai*
Whereas [the sages] (believe) is of [rain from] crops to destroy reach be
SRL: **A1** **A1**
Gold: **A0** **A1**
(Whereas the sages believe that crops are destroyed due to the rains)

(3) *lekin [unhe] is baat ka [afsos] (hai)$_V$*
But [they] this thing of [regret] (be)
SRL: **A1** **A1**
Gold: **A1** **A2-ATR**
(But they regret this)

SRL labels in above examples are from Anwar and Sharma's model [2]. Unlike their system, our model is able to label all of the above examples correctly which is equivalent to the Gold labels. In sentence 1, "ek aur kaidi ne" is labeled as ARG1 whilst it can be clearly seen that it reflects agentivity. Since we use 'vibhakti' as a feature ('ne' in this case), our system labels it as ARG0. In the second sentence, "rishi vaigyanik" is a NP chunk which can be labeled as either ARG0 or ARG1 depending on where it is getting used. In some sentence, it could have been used as ARG1 also, unlike ARG0 in this case. This tells us how necessary it is to use more features which describe the context or the neighbouring tokens/nodes in the sentence and in the syntactic parse tree. Using the dependency path and pre-trained word embeddings solves such problems to some extent. The word embeddings are trained on a very large dataset of natural language text which learns the representation for each token based on the tokens occurring before and after itself. Similar explanation could be given for example 3.

Now, we look at the confusion matrix (Table 4.9) of Hindi test data generated by our model and analyze the errors that still remain. Even though our argument identification step gains significantly over the previous baseline but we can still see many chunks which are non-arguments but getting some label and similarly many arguments being labeled as non-arguments. Though there is still some confusion in 'ARG0' and 'ARG1' but it is significantly reduced. We still see 'A2-ATR' labeled as 'A1' but lesser than previous model. Let us look at the examples below.

	NULL	**A0**	**A1**	**A2**	**A2-ATR**	**A2-GOL**	**A2-LOC**	**A2-SOU**	**AM-ADV**	**AM-DIR**	**AM-LOC**	**AM-MNR**	**AM-MNS**
NULL	0	160	357	6	44	0	0	3	5	1	142	46	1
A0	116	173	29	1	0	0	0	0	1	0	0	0	0
A1	254	10	276	3	7	0	0	0	0	0	4	4	1
A2	17	1	4	10	0	0	0	0	0	0	1	0	0
A2-ATR	22	0	10	0	40	0	0	0	1	0	0	1	0
A2-GOL	7	0	0	0	0	4	0	0	0	0	2	0	0
A2-LOC	3	0	0	0	0	0	0	0	0	0	7	1	0
A2-SOU	9	0	0	0	0	0	0	3	1	0	1	0	0
AM-ADV	14	0	1	0	1	0	0	1	6	0	4	2	0
AM-DIR	3	0	0	0	0	0	0	1	1	2	0	1	0
AM-LOC	68	1	2	0	0	0	0	2	1	0	131	3	0
AM-MNR	42	1	2	1	4	0	0	1	1	0	11	38	1
AM-MNS	4	0	1	0	0	0	0	0	0	0	0	2	4

Table 4.10 Confusion Matrix for Hindi test data with our model

(4) *yeh barish* [*maansuni hawaon ke aage*] *(chalne)*$_{\mathbf{V}}$ *wali* [*hawaon ki*] *sakriyata ke kaaran hoti hai*

This rain [monsoon winds in front of] (going) one [winds of] activity of reason be

SRL: **AM-LOC** **A1**

Gold: **AM-MNR** **A1**

(These rains are due to the active winds following the monsoon winds)

(5) *Saraimeer, Baskhari ke bhi [kai log] inke [jaal mein] (fans)$_V$ chuke hain*

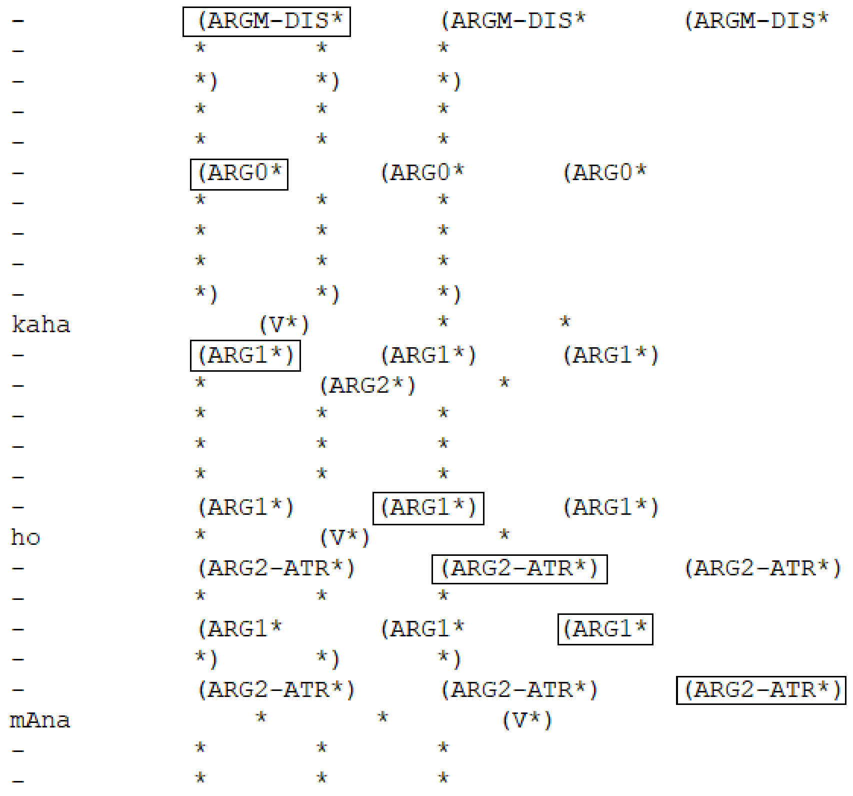
Saraimeer, BasKhari of too [many people] their [trap in] (fall) be

SRL: **A1** **AM-LOC**

Gold: **A1** **A2-LOC**

(People of Saraimeer, Baskhari have also fallen into their trap)

In sentence 4, we can see that “maansuni hawaon ke aage” is labeled as AM-LOC which is not correct. This is probably due to the occurrence of “ke aage” in the phrase which implies the presence of locative information but in the context of this sentence that is not the case. ‘ARG2-LOC’ is very often assigned the role ‘ARGM-LOC’, one example of which can be seen in sentence 5. This probably happens because both convey locative information but ARGM-LOC refers to physical location and ARG2-LOC refers to abstract location.



```
-            (ARGM-DIS*      (ARGM-DIS*       (ARGM-DIS*
             *       *       *
             *)      *)      *)
-            *       *       *
-            *       *       *
-            (ARG0*      (ARG0*       (ARG0*
-            *       *       *
-            *       *       *
-            *       *       *
-            *)      *)      *)
kaha             (V*)        *        *
-            (ARG1*)     (ARG1*)      (ARG1*)
-            *       (ARG2*)      *
-            *       *       *
-            *       *       *
-            *       *       *
-            (ARG1*)     (ARG1*)      (ARG1*)
ho           *       (V*)         *
-            (ARG2-ATR*)     (ARG2-ATR*)      (ARG2-ATR*)
-            *       *       *
-            (ARG1*      (ARG1*       (ARG1*
-            *)      *)      *)
-            (ARG2-ATR*)     (ARG2-ATR*)      (ARG2-ATR*)
mAna             *       *        (V*)
-            *       *       *
-            *       *       *
```

Figure 4.1 Sample prediction for a sentence in Hindi test data using our model

A big problem with both Anwar and Sharma's [2] model and our model occurs in the case when multiple predicates are present in a sentence. Almost 60% sentences have more than 1 predicate and it is very often seen in these sentences that a chunk is labeled as the same argument with respect to every predicate. We can see from Figure 4.1 that almost all chunks are given the same label with respect to each predicate. The labels in the boxes represent the actual labels which are in the gold data. All other labels are incorrectly assigned by our system.

4.6 Conclusion

In this chapter, we presented a statistical semantic role labeler for Hindi/Urdu Propbank data which performs better than the previous models for these languages. We discussed about the features we used, some taken from the existing baseline and the rest introduced by us for the first time for Hindi/Urdu SRL. Then we saw the contributions by each of the new feature and the overall improvements over the previous baseline. Finally, we look at the errors made by the previous baseline system and how we correct them.

We also report the limitations in our model. We make another effort to further improve the results and reduce errors by proposing various deep learning models in the next chapter. We also show the analysis of the deep learning models with respect to this model.

Chapter 5

Exploring Deep learning methods for Semantic role labeling in Indian Languages

5.1 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 a sentence given as the input. To make this process fully automatic, we need a program which must take a sentence in its raw form and then output the most probable predicate-argument structure without any other inputs being provided. To do this, we propose the use of deep learning which tries to learn its own representation of the sentence and then based upon it predict the predicate-argument structure.

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 by Anwar and Sharma [2]. Following which another system was seen which is depicted in the previous chapter. Both these systems are based on traditional approaches towards SRL as seen in Gildea and Jurafsky's [22] and Xue and Palmer's [70] work.

We have seen in the discussion in the last chapter that even though our statistical model performs better than the previous baseline but it still fails to capture details such as (a) accurately identifying whether a chunk is an argument, (b) confusion amongst the labels and (c) labeling a chunk as the same argument irrespective of the predicate and its context which makes it an inadequate model (Figure 4.1). Chunks being given the same label always is probably because their is not enough learning from the contextual features. Moreover, this occurs when the sentences have multiple predicates and are larger in length. In this chapter, we attempt to solve these problems by using state-of-the-art approaches which have proven to perform better or at par with the traditional methods. Zhou and Xu [74], He et al. [27], and Marcheggiani et al. [38] have shown that deep neural networks perform better than state-of-the-art role labelers even without using any syntactic information. They have published their results for English propbank which is a large data bank. These networks have their edge over the traditional models when labeling distant phrases because they are able to remember the context much better (LSTM Network).

In this work, we try to apply various neural networks for Indian language SRL and analyze if they are able to learn more from the context and whether they work for such low data settings. 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 first perform, (1) Argument Identification i.e., binary classification and then (2) 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 [53]. 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 in earlier systems [55]. This model performs slightly better than the model from previous chapter. 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 by Anwar and Sharma [2] 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 a few features we talked about in the chapter 4, 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.

5.2 Models

5.2.1 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 [26], 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 [70]. 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.

5.2.2 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 matrices 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:

$$y_m^{(t)} = \sigma(\sum_f w_f^m x_f^{(t)} + \sum_i w_i^m y_i^{(t-1)}) \tag{5.1}$$

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 RNN's variant, the LSTM network because it is known to handle long range dependencies [74] 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 [6]. To resolve these issues, long short-term memory [30] was presented.

LSTM Network. Long short-term memory (LSTM) [30, 24] 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 the three activated multiplicative gates: the input gate, the forget gate and the output gate. These changes improve the RNN model for sequence modeling. Figure 5.1 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 cell's state value for block m. Number of cells in a block is fixed to be one.

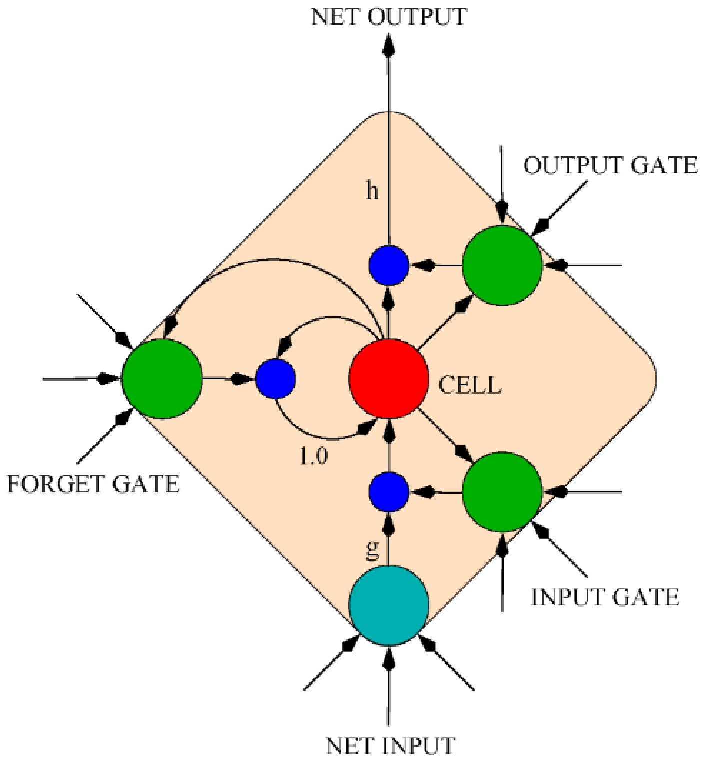


Figure 5.1 LSTM memory block with one cell. [24]

α, β and γ stand for the input, forget and output gates activation values. All three multiplicative gates have different activation σ respectively and the computations are done as follows:

$$
\begin{aligned}
h_m^{(t)} &= \sigma_h(\sum_f w_{f,h}^m x_f^{(t)} + \sum_i w_{i,h}^m y_i^{(t-1)}) \\
\alpha_m^{(t)} &= \sigma_\alpha(\sum_f w_{f,\alpha}^m x_f^{(t)} + \sum_i w_{i,\alpha}^m y_i^{(t-1)} + w_\alpha^m c_m^{(t-1)}) \\
\beta_m^{(t)} &= \sigma_\beta(\sum_f w_{f,\beta}^m x_f^{(t)} + \sum_i w_{i,\beta}^m y_i^{(t-1)} + w_\beta^m c_m^{(t-1)}) \\
\gamma_m^{(t)} &= \sigma_\gamma(\sum_f w_{f,\gamma}^m x_f^{(t)} + \sum_i w_{i,\gamma}^m y_i^{(t-1)} + w_\gamma^m c_m^{(t)})
\end{aligned}
\tag{5.2}
$$

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 cell's 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 [59]. 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 [74].

5.2.3 Model A - Path embedding model

This model closely follows the architecture of PathLSTM [55] and is shown in Figure 5.2. 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 predicate's chunk. The head-word embeddings are pre-trained embeddings computed similarly as in previous chapter. 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 our work shown in previous chapter because there we 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[*] 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 [55]. Thus, some representation learning should be done, instead of taking the full path as it is.

The dependency path is given to the 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 block's output state from the network gives us the embedding of this path. As shown in Figure 5.2, the embedding of a dependency path, specifically is h_n which is returned by the LSTM layer's 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 [55].

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

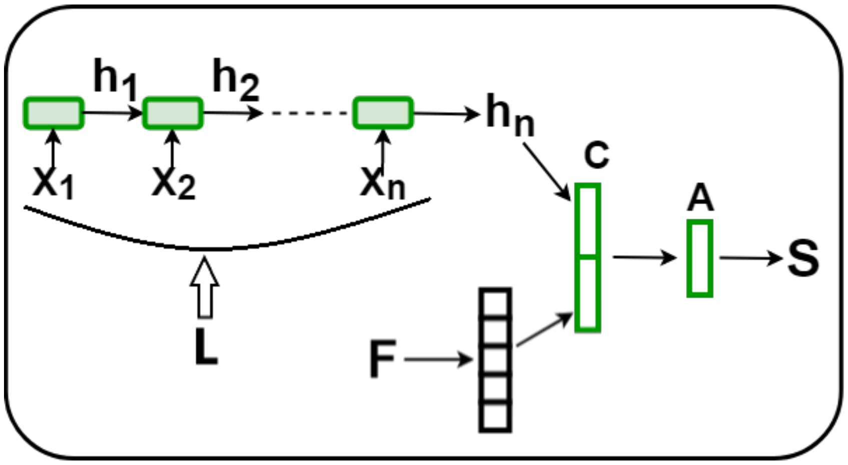


Figure 5.2 Model A - Path embeddings with LSTM

Model A is depicted in Figure 5.2 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 block's 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 [70] as well as for Hindi and Urdu [2]. This also means that different path embeddings are learned depending on what the task is. The features F are taken as it is from what we proposed in chapter 4, 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 chunk's vibhakti (post-positional), head word POS tag, candidate chunk's POS category.

5.2.4 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 $O(n_pL)$. At each step in the sequence, the current word's 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 [20, 74] 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$_\text{k}$' - is given to words at beginning of the chunks and 'I$_\text{k}$' - is given to the words inside the chunks. k denotes the various roles shown in chapter 3. 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: $\sum_{t=1}^{n} \mathbf{log} p(y_t|s)$. To incorporate constraints like IOB order, structural constraints (explained later in this section), we extend the scoring function [27] 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 5.3. 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 5.4. Recent best works [74, 38, 27] 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 [27]. 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$_\text{k}$ tag followed by I$_\text{k}$ tag or say I-ARG0 followed by B-ARG1 etc. We apply infinitely high penalty for such transitions.

SRL Constraints: Some structural constraints have been applied for English SRL [53, 66]. 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.

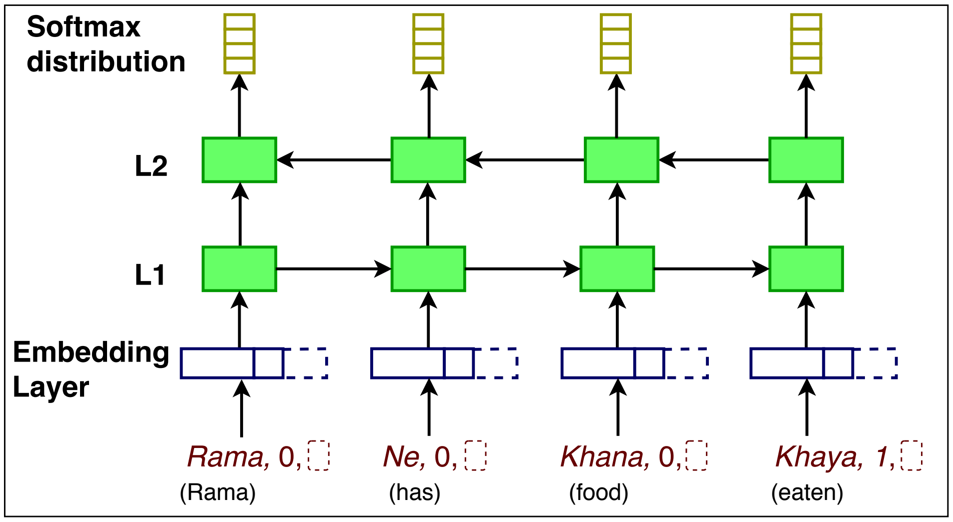


Figure 5.3 Model B and C - 2 Layer BiLSTM model.

5.2.5 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 Anwar and Sharma [2] and our statistical model in the last chapter, we see the effect of adding a few important features like dependency label, chunk category, parent's category, head word of the chunk and POS tags of tokens.

In Figure 5.3, the red colored input with dotted box is meant to incorporate any other features and this is where we first experiment by adding one feature at a time and look at its effect on results. In the final model, we combine all the features which showed an increment on top of Model B. 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, chunk category, parent category and head word. POS tags remain unique for each word.

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

5.3 Experiments

Evaluation metrics. To compare Model A with the work in previous chapter (call it **Model 0**), we have used the same evaluation metric as used for that model, 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 in CoNLL-2005 shared task [16]. 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. This was also depicted in Figure 2.1. 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.
As an exceptional case, the verb argument of each proposition is excluded from the evaluation in the script. In the Hindi/Urdu data, exactly every time, the verb corresponds to the target verb of the proposition, which is provided as input. Except for non-trivial cases, this situation makes the verb fairly easy to identify and, since there is one verb with each proposition, evaluating its recognition over-estimates the overall performance of a system. For this reason, the verb argument is excluded from evaluation. Since we don't have a development/validation set, we optimize our system's parameters based on the test set.

5.3.1 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 chapter's work 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 the corresponding results from Model 0 in Table 5.1 and Table 5.2 for identification and classification respectively.

Language	Model	Precision	Recall	F1-score
Hindi	Model 0	91.41	90.49	90.94
	Model A	**93.35**	**93.29**	**93.32**
Urdu	**Model 0**	**92.05**	**91.49**	**91.76**
	Model A	91.50	91.17	91.33

Table 5.1 Argument Identification results.

Language	Model	Precision	Recall	F1-score
Hindi	Model 0	65.04	66.62	65.80
	Model A	**70.23**	**72.25**	**71.22**
Urdu	**Model 0**	**86.72**	**86.37**	**86.54**
	Model A	85.57	85.52	84.73

Table 5.2 Argument Classification results.

(* Model 0 represents our statistical model from the previous chapter)

5.3.2 Model B

Our BiLSTM has only 2 layers - 1 forward LSTM and 1 backward LSTM. The hidden dimension is set to be 300. A softmax layer predicts the output distribution at each token. All weight matrices of the BiLSTM are initialized as random orthonormal matrices as described in Saxe et al.'s work [57]. The pre-trained embeddings for both Hindi and Urdu are taken from Fast-Text [14]. 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 [73] 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 don't have a development data to check the development loss. In the final model we only use the IOB constraints.

Language	Model	Precision	Recall	F1-score
Hindi	Model 0	42.52	49.66	45.81
	Model A	44.38	50.53	47.26
	Model B	**56.71**	**56.39**	**56.55**
Urdu	**Model 0**	**86.41**	**67.16**	**75.58**
	Model A	85.01	66.91	74.88
	Model B	63.10	63.20	63.15

Table 5.3 SRL full task results evaluated using conll-2005 shared task evaluation script.
(* Model 0 represents our statistical model from the previous chapter)

5.3.3 Model C

We train Model C exactly in the same way as Model B. The only difference is that this time we experiment by adding the syntactic features we talked about in section 5.2.5. We experiment by adding only a single feature at once. We can see from Table 5.4, that for every feature except 'Head word', there is a gain in results over Model B. Therefore, we make a final model adding all features which show a gain in performance (except Head-word) and call it Model C. We show the final results in Table 5.5.

	Precision	Recall	F1-score
Model B	56.71	56.39	56.55
+ *Dependency label*	70.41	70.41	70.41
+ *Chunk category*	61.17	59.85	60.51
+ *Parent chunks' category*	64.61	64.61	64.61
+ *Head word*	50.07	45.27	47.55
+ *POS tag*	65.23	64.30	64.76
+ *All Features - Head word*	**74.14**	**72.95**	**73.54**

Table 5.4 Impact on results by adding features to Model B for Hindi test data

5.4 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 Table 3.2 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. Vaidya et al. [69] also showed that whenever 'pof' dependency/karaka relation is seen it almost always falls into this category.

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 work in previous chapter. 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

Language	Model	Precision	Recall	F1-score
Hindi	Model 0	42.52	49.66	45.81
	Model A	44.38	50.53	47.26
	Model B	56.71	56.39	56.55
	Model C	**74.14**	**72.95**	**73.54**
Urdu	Model 0	86.41	67.16	75.58
	Model A	85.01	66.91	74.88
	Model B	63.10	63.20	63.15
	Model C	**79.66**	**80.12**	**79.88**

Table 5.5 SR full task results for all models we have proposed
(* Model 0 represents our statistical model from the previous chapter)

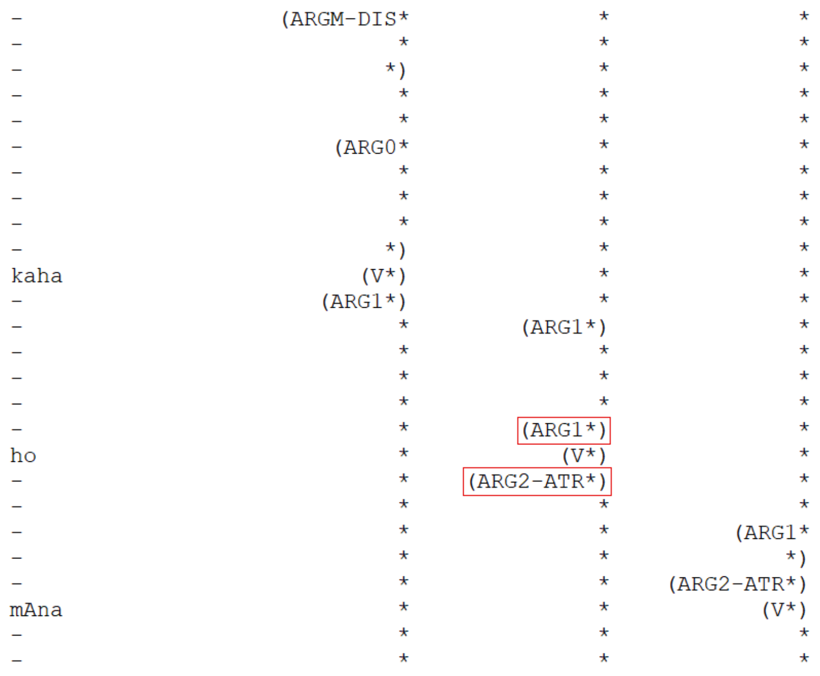
performance. The primary reason for this could be attributed to less training points in data. We only use IOB constraints since they gave a significant improvement in results whilst the Unique Core Roles constraint did not. These models are overall the best performers as they are in a way learning more complex features from the whole sentence than the pre-defined features used in model A.

Table 5.6 shows the confusion matrix for Model C on Hindi test data. We see a very high reduction in the number of chunks which are NULL but labeled otherwise and vice-versa. We can see that the confusion between 'ARG0' and 'ARG1' has also considerably reduced relative to the number of arguments classified correctly (which have increased substantially).

	NULL	A0	A1	A2	A2-ATR	A2-GOL	A2-LOC	A2-SOU	AM-ADV	AM-DIR	AM-LOC	AM-MNR	AM-MNS
NULL	0	45	42	1	3	1	2	1	7	2	10	18	0
A0	51	242	21	2	3	1	0	0	0	0	0	0	0
A1	46	14	478	1	9	1	1	0	0	0	5	2	1
A2	2	2	4	21	0	0	0	0	0	0	3	1	0
A2-ATR	7	1	17	0	48	0	0	0	1	0	0	2	1
A2-GOL	0	0	2	0	0	6	1	0	0	0	4	0	0
A2-LOC	0	0	2	0	1	0	0	0	0	0	8	0	0
A2-SOU	3	0	1	0	0	0	1	3	0	0	5	0	0
AM-ADV	5	0	0	0	1	0	1	0	13	0	4	6	0
AM-DIR	1	1	0	0	0	0	0	0	1	1	1	2	0
AM-LOC	23	2	5	1	2	1	0	0	0	0	167	4	2
AM-MNR	14	0	3	2	8	0	0	1	6	0	7	53	0
AM-MNS	3	0	1	0	0	0	0	0	0	0	0	4	5

Table 5.6 Confusion Matrix for Hindi test data with Model C

We can see that ‘ARG2-ATR’ is still getting labeled as ‘ARG1’ many times even with this model. This can be attributed to the fact that ‘ARG2-ATR’ (attribute) and ‘ARG1’ (patient) are sometimes very confusing which can be seen in example shown below. Let us look at the example below to look the improvements and limitations in our model.



```
-         (ARGM-DIS*             *             *
-                  *             *             *
-                 *)             *             *
-                  *             *             *
-                  *             *             *
-              (ARG0*            *             *
-                  *             *             *
-                  *             *             *
-                  *             *             *
-                 *)             *             *
kaha             (V*)            *             *
-             (ARG1*)            *             *
-                  *        (ARG1*)            *
-                  *             *             *
-                  *             *             *
-                  *             *             *
-                  *        (ARG1*)            *
ho                 *          (V*)             *
-                  *    (ARG2-ATR*)            *
-                  *             *             *
-                  *             *         (ARG1*
-                  *             *            *)
-                  *             *   (ARG2-ATR*)
mAna               *             *          (V*)
-                  *             *             *
-                  *             *             *
```

Figure 5.4 Sample prediction for a sentence in Hindi test data using Model C

Figure 5.4 above represents the same sentence shown in Figure 4.1 of the previous chapter. We can clearly see that this model does not repeat the same behaviour as the statistical which was classifying the chunks with the same label for every predicate. The labels marked with red colour are incorrectly classified. The actual labels for this sentence with respect to the predicate 'hai' are shown below.

(1) *Iss beech, Tehelka ke pradhan sampadak Tarun Tejpal ne kaha ki* [*unhe*]$_{\textbf{ARG1}}$ *iss baat ki* [*raahat*]$_{\textbf{A2-ATR}}$ *(hai)*$_{\textbf{V}}$ [*ki*]$_{\textbf{AM-MNR}}$ *unke rukh ko sahi maana gaya*
(Meanwhile, the chief editor of Tehelka Tarun Tejpal said he is relieved that his line was understood clearly)

We see that "raahat" is labeled as 'ARG1' by our system while it is 'ARG2-ATR' in the gold data. Similarly, "ki" is labeled as 'A2-ATR' by our system but is actually 'AM-MNR'. These labels are difficult to label even for a human annotator and have only a few occurrences in the training data due to which these classes are not well trained. For Urdu, we see that training data is overall very low and hence Model B is not able to perform very well. We need to add features to improve the results in case of Urdu.

For analyzing our network based on how good it does to capture long range dependencies, we perform two types of analysis - (1) We find the F1-score of our model for different distance of the arguments with respect to the predicate. (2) We compute the F1-score for various length of sentences of the test data. To depict (1), we plot a graph shown in figure 5.4. Model 0 in the legend represents the statistical model from previous chapter. Here we compared only the best and the worst performing models (Model 0 and Model C) to easily capture the details. It is evident from the plot that performance degrades with distance for both the models. Model C performs very well for small distances but we also see gain for higher distances like 10-15 and especially more than 20.

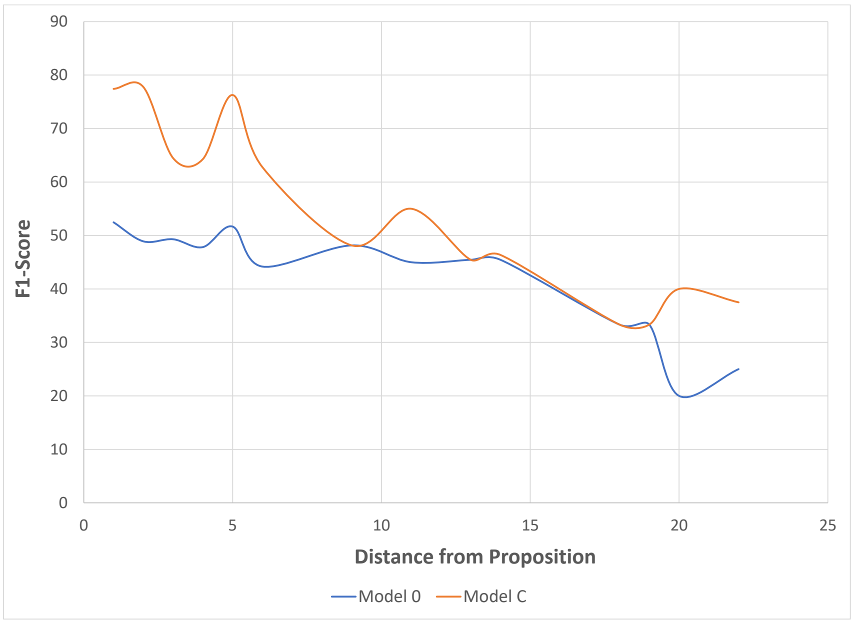


Figure 5.5 F1 score by distance between arguments and predicate for Hindi test data.

For (2), we plot figure 5.5. It compares all 4 models proposed by us in this thesis by presenting F1 scores at different sequence lengths on Hindi Propbank test data. We did not take an average of the results at each data point because of the very low size of data and hence we don't comment on whether the results of SRL improve or deteriorate with distance. For example, there is a single sentence of length 54 in the test set and since its F1 is 100 for Model C, it does not mean that all sentences of this length will perform good. So we can't really comment on the performance with respect to distance. Usually, the results degrade with increasing distance as seen in most of the early work. However, we can say for sure that each model is better in performance than the previous model(s) and it is more evident at longer distances which proves that deep learning models are better at learning long range dependencies even when trained on a smaller training data as in Hindi/Urdu.

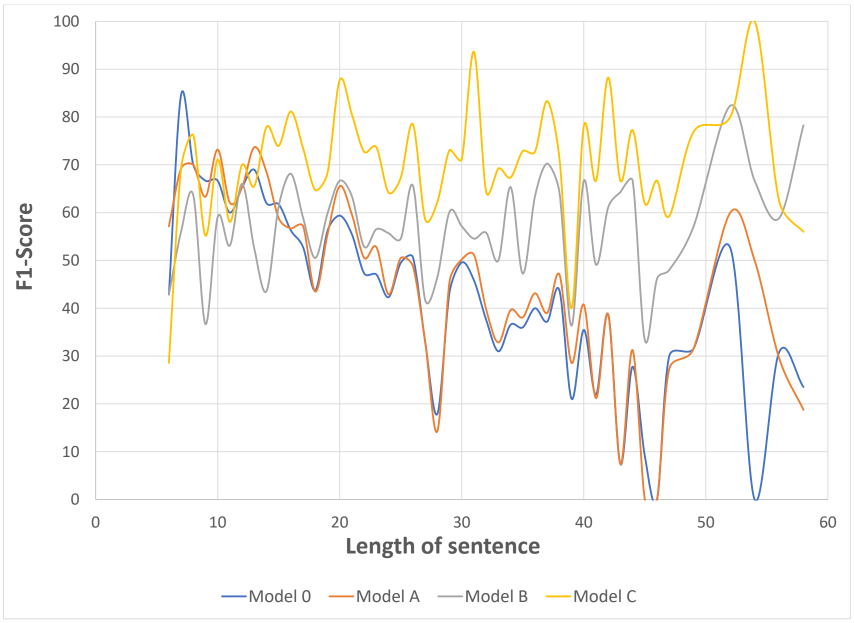


Figure 5.6 F1 score by Sentence length for Hindi test data.

5.5 Conclusion

In this chapter, we presented three deep learning models all of which are based on LSTM. We compare our first model with the model presented in previous chapter and see that it performs slightly better for Hindi and nearly as good for Urdu. We have proposed a new baseline system for Indian languages.
Next, we propose the use of CONLL-2005 shared task [16] based evaluation and give the results based on that for all 4 models as presented in this thesis.

Chapter 6

Conclusions and Future Work

In this thesis, we have described our efforts in improving Semantic role labeling for Indian languages. We achieved this by creating four types of models using different approaches like traditional machine learning methods, relatively newer deep learning neural networks and a mix of these two methodologies. All four models outperform the existing baseline for both Hindi and Urdu.

We began by providing a detailed coverage on all of the work that has been used to perform semantic role labeling from the very start up until now. This covers every aspect such as (1) the features that have been discovered as being important in identification and classification, (2) the various traditional machine learning approaches that have been proposed to learn semantic role labeling, (3) inferencing methods to further improve and globalize the results, (4) a look at different types of labeling tasks and how evaluation is done for them, (5) the progress made by deep learning in this field and finally we look at (6) the prior work on SRL for ILs. Also, for the first time, a detailed description and analysis of the Hindi/Urdu Propbank data have been provided by us. We also talk about the various kinds of datasets that are used for other languages as well.

We proposed some new features to train a statistical model similar to prior work and our system made performance gains not only in identification of arguments but also in classification for both Hindi and Urdu. Our classification was able to improve from 49 F1 score to 65.8 F1 for Hindi using a Support Vector Machine Classifier. We depicted the gain with every feature we added on top of the last baseline. We carried out 5-fold cross validation to verify the consistency in train:test split of data.

Finally, we built 3 deep learning models, all using LSTM. The first model aims to improve feature representation by embedding path using a LSTM. The second model goes fully syntax-agnostic and learns the vector representation for each token in a sentence in regard to the end task of role labeling. The final model adds few features on top of the second model. We also shift to CONLL standard for evaluation. These experiments showed that the final model is able to achieve the best F1 score of 73.54 for Hindi and 79.88 for Urdu. We give an analysis to reason out the performance of these models.

We have also released the names of training and testing data files used by us in all the experiments in Section 3.4, which would be extremely useful for the researchers working on semantic analysis for

Indian languages. Some of these are not available publicly and can be sourced by contacting the author of the respective datasets.
As per our conclusion, though we have created a strong baseline, we still believe that there is a lot of scope for further research and improvement in this area for the following reasons.

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 could give more reasons for 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.

Performance gain is quite impressive using deep learning models and it is motivating to see that syntax-agnostic models or models relying on minimal syntax can outperform traditional methods which heavily rely on the previous step, i.e., syntactic parsing. However, we see that when we add some syntactic feature to a syntax free model, there is a huge gain in performance. This manifests potential for high grade parsers to further improve deep learning role labelers.

Semantic role labeling is an intermediate step in solving real world problems like machine translation, question-answering etc. Hence, exploration can be done to make improvements in SRL only systems, joint learning systems or creation of such a deep learning system to directly model the end application directly.

Related Publications

- Aishwary Gupta, Manish Shrivastava : Enhancing Semantic Role Labeling in Hindi and Urdu. Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)
- Aishwary Gupta, Akshay Pawale, Manish Shrivastava : Deep Learning methods for Semantic Role Labeling in Indian Languages. Proceedings of ICON-2018, Patiala, India. December 2018, pages 59 to 68

Bibliography

[1] M. Anwar, R. A. Bhat, D. Sharma, A. Vaidya, M. Palmer, and T. A. Khan. A proposition bank of urdu. In N. C. C. Chair), K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, and S. Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may 2016. European Language Resources Association (ELRA).

[2] M. Anwar and D. Sharma. Towards building semantic role labeler for indian languages. In N. C. C. Chair), K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, and S. Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may 2016. European Language Resources Association (ELRA).

[3] C. F. Baker, C. J. Fillmore, and J. B. Lowe. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics, 1998.

[4] 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.

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

[6] 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.

[7] A. Bharati. Anncorra: Annotating corpora guidelines for pos and chunk annotation for indian languages. 2006.

[8] A. Bharati, V. Chaitanya, R. Sangal, and K. Ramakrishnamacharyulu. *Natural language processing: a Paninian perspective*. Prentice-Hall of India New Delhi, 1995.

[9] A. Bharati, R. Sangal, and D. M. Sharma. Ssf: Shakti standard format guide. 2007.

[10] A. Bharati, S. Venkatapathy, and P. Reddy. Inferring semantic roles using sub-categorization frames and maximum entropy model. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 165–168. Association for Computational Linguistics, 2005.

[11] R. Bhatt, B. Narasimhan, M. Palmer, O. Rambow, D. M. Sharma, and F. Xia. 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, 2009. Association for Computational Linguistics.

[12] A. Björkelund, L. Hafdell, and P. Nugues. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 43–48. Association for Computational Linguistics, 2009.

[13] H. C. Boas. Bilingual framenet dictionaries for machine translation. In *LREC*, 2002.

[14] 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.

[15] X. Carreras and L. Màrquez. Introduction to the conll-2004 shared task: Semantic role labeling. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, 2004.

[16] 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.

[17] J. Chen and O. Rambow. Use of deep linguistic features for the recognition and labeling of semantic arguments. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, 2003.

[18] 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.

[19] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[20] 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.

[21] 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.

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

[23] D. Gildea and M. Palmer. The necessity of parsing for predicate argument recognition. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 239–246. Association for Computational Linguistics, 2002.

[24] 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.

[25] K. Hacioglu. Semantic role labeling using dependency trees. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1273. Association for Computational Linguistics, 2004.

[26] 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.

[27] 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.

[28] K. M. Hermann, D. Das, J. Weston, and K. Ganchev. Semantic frame identification with distributed word representations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1448–1458, 2014.

[29] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

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

[31] S. Husain, P. Mannem, B. R. Ambati, and P. Gadde. The icon-2010 tools contest on indian language dependency parsing. *Proceedings of ICON-2010 Tools Contest on Indian Language Dependency Parsing, ICON*, 10:1–8, 2010.

[32] J. D. Hwang, A. Bhatia, C. Bonial, A. Mansouri, A. Vaidya, N. Xue, and M. Palmer. Propbank annotation of multilingual light verb constructions. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 82–90. Association for Computational Linguistics, 2010.

[33] P. Kingsbury and M. Palmer. From treebank to propbank. In *LREC*, pages 1989–1993. Citeseer, 2002.

[34] P. Koomen, V. Punyakanok, D. Roth, and W.-t. Yih. 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, 2005.

[35] T. Lei, Y. Zhang, L. Marquez, A. Moschitti, and R. Barzilay. High-order low-rank tensors for semantic role labeling. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1150–1160, 2015.

[36] B. Levin. *English verb classes and alternations: A preliminary investigation*. University of Chicago press, 1993.

[37] 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.

[38] 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.

[39] M. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

[40] L. Màrquez, P. Comas, J. Giménez, and N. Catala. Semantic role labeling as sequential tagging. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 193–196. Association for Computational Linguistics, 2005.

[41] R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics, 2005.

[42] G. Melli, Y. Wang, Y. Liu, M. M. Kashani, Z. Shi, B. Gu, A. Sarkar, and F. Popowich. Description of squash, the sfu question answering summary handler for the duc-2005 summarization task. *safety*, 1:14345754, 2006.

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

[44] A. Moschitti, A.-M. Giuglea, B. Coppola, and R. Basili. Hierarchical semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 201–204, 2005.

[45] S. Narayanan and S. Harabagiu. Question answering based on semantic structures. In *Proceedings of the 20th international conference on Computational Linguistics*, page 693. Association for Computational Linguistics, 2004.

[46] J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. The conll 2007 shared task on dependency parsing. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007.

[47] N. E. Ozgencil and N. McCracken. Semantic role labeling using libsvm. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 205–208. Association for Computational Linguistics, 2005.

[48] 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.

[49] S. P. Ponzetto and M. Strube. Semantic role labeling using lexical statistical information. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 213–216. Association for Computational Linguistics, 2005.

[50] S. P. Ponzetto and M. Strube. Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, 2006.

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

[52] S. S. Pradhan, W. H. Ward, K. Hacioglu, J. H. Martin, and D. Jurafsky. Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, 2004.

[53] 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.

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

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

[56] M. Roth and K. Woodsend. Composition of word representations improves semantic role labelling. In *EMNLP*, pages 407–413, 2014.

[57] 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.

[58] K. K. Schuler. Verbnet: A broad-coverage, comprehensive verb lexicon. 2005.

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

[60] 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.

[61] V. Srikumar and C. D. Manning. Learning distributed representations for structured output prediction. In *Advances in Neural Information Processing Systems*, pages 3266–3274, 2014.

[62] M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. Using predicate-argument structures for information extraction. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, 2003.

[63] M. Surdeanu, R. Johansson, A. Meyers, L. Màrquez, and J. Nivre. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177. Association for Computational Linguistics, 2008.

[64] M. Surdeanu, L. Màrquez, X. Carreras, and P. R. Comas. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*, 29:105–151, 2007.

[65] 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.

[66] 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.

[67] K. Toutanova, A. Haghighi, and C. D. Manning. Joint learning improves semantic role labeling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 589–596. Association for Computational Linguistics, 2005.

[68] K. Toutanova, A. Haghighi, and C. D. Manning. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191, 2008.

[69] 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.

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

[71] S.-T. Yi, E. Loper, and M. Palmer. Can semantic roles generalize across genres? In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 548–555, 2007.

[72] S.-t. Yi and M. Palmer. The integration of syntactic parsing and semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 237–240, 2005.

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

[74] 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.