Chapter 4

Hindi Compound Noun Interpretation Using Machine Learning

4.1 Introduction

The previous chapter presented a detailed linguistic analysis of compound noun interpretation, we have presented semantic relation set, annotation schema, compound noun bracketing for Hindi language. This chapter will describe the process of creating compound noun dataset to use in machine learning experiments as well as discuss the result of different experiments done on this dataset.

The semantic interpretation of compounds is essential in various NLP tasks. This task deals with identifying the semantic relations between the constituents of Compound Nouns. For example, in the compound noun *butterknife*, knife is used for putting butter on bread, therefore, the relation between the constituent is termed as Purpose relation. But in *glass knife*, knife is made up of the material glass and

the relation is Made of relation. Humans can understand the relations between the noun pairs through their world knowledge. However, providing this knowledge to a machine is a herculean task. Recent research trends in computational semantics have used word embeddings, Vector Space representation, machine learning, and deep neural network classifiers for many NLP tasks including semantic similarity, compound noun interpretation, relation extraction etc. and got significant results for the compound noun interpretation tasks. The interpretation of compound nouns using abstract semantic relations worked on the hypothesis that if different semantic relations found between the constituents of the compound is identified, then the meaning of that compound can also be identified. In this chapter we have used this approach and aimed to interpret the semantic of compound nouns by classifying the different semantic relations using machine learning algorithms and checking the accuracy of different models on the compound noun interpretation task.

The approaches used in the chapter comprise machine learning techniques to treat compound interpretation as a machine learning classification task using a standard set of semantic relations as the prediction labels. Any computational approach often requires a large training dataset or corpus with a good distribution across all types of semantic relations and support of an underlying lexical knowledge base to extract features of the nouns. In our work, we could use only Hindi WordNet as a supporting lexical database. Moreover, the training data set also contained only certain kinds of relations repeatedly. For the classification task, first we need to prepare a dataset of Hindi compound nouns with proper annotation of the relation. Then the selection of the appropriate classifier and the adequate features are important steps. We will present the detailed experiment in the present chapter and discuss the result with analysis.

The chapter is divided into two main sections: First part presents the compound noun interpretation using SVM and Random Forest machine learning algorithms

and Noun features. The second part shows the experiment using word embedding features and SVM and BERT classifiers

4.2 Machine Learning and NLP

Statistical analysis of langauges have been in the dominant area of study in the last several decades starting from rule based and knowledge based to simple statistical methods in basic probability theory and now the machine learning and deep learning approaches for solving various NLP tasks and proposing computational theory of langauge Manning (2000); Khan et al. (2016). Some of the models are: hidden Markov models (HMM), conditional random field (CRF), maximum entropy models (MaxEnt), support vector machines (SVM), Naïve Bays, and deep learning (DL). Machine Learning has taken a front stage in the NLP research in the past few years because of the availability of the large corpus as well as the increased efficiency of the modern computers.

4.3 Theoretical Background of Machine Learning

This section provides a theoretical background for the classification of semantic relations between the constituents using machine learning methods. The section presents the performance metrics used for the evaluation of the output from models and gives a brief demonstration of the models SVM with Kernels, Random forest and BERT classifiers. We have used Embeddings (Word2Vec and Bert Embedding) as features for the model. The theory behind this is also presented in this section.

4.3.1 Performance metrics

The effectiveness of any machine learning model is checked with some metric based on test datasets. After training, testing and getting the output using a machine learning model the next step is to evaluate the performance of the model. Different performance metrics are used to test the performance of the machine learning model. The selection of suitable performance metrics vary depending on the problems. The study used a machine learning task as a classification task. For classification problems generally, Confusion matrix, accuracy score as well as Precision, Recall, and F score performance metrics are used. Performance metrics for classification tasks evaluate the model and tell how good or bad the output is, but each of them do it in a different way. The description of the performance metrics used in the study is provided here.

4.3.1.1 Confusion Matrix

Confusion matrix is one of the most used and intuitive metrics for evaluating the performance of the model. It is a measure of how correctly and accurately a classifier can classify the instances into two or more classes. Confusion matrix is the tabular visualization form of the actual value and predicted value of the model. The output is of the form (NxN matrix). The rows of the confusion matrix represent the instances in a predicted class and the columns represent the instances in an actual class. On the basis of confusion matrices other metrics evaluate the results of the model.

A simple performance measure for binary classification using confusion metrics is given in the table 4.4 below:

	Predicted Class		
	Positive	Negative	Total
Positive	TP	FN	P
Negative	FP	TN	N

TABLE 4.1: Confusion Matrix

True Positives (TP): True positives are the cases when the actual class of the data point was 1(True) and the predicted is also 1(True)

True Negatives (TN): True negatives are the cases when the actual class of the data point was 0(False) and the predicted is also 0(False)

False Positives (FP): False positives are the cases when the actual class of the data point was 0(False) and the predicted is 1(True). False is because the model has predicted incorrectly and positive because the class predicted was a positive one. (1)

False Negatives (FN): False negatives are the cases when the actual class of the data point was 1(True) and the predicted is 0(False). False is because the model has predicted incorrectly and negative because the class predicted was a negative one. (0)

Accuracy Accuracy measure is the simplest metric used for performance of the model. The accuracy score is calculated by dividing the number of correct predictions to the number of total predictions multiplied by 100.

$$Accuracy = \frac{Correct\_predictions}{total\_number\_of\_predictions} \tag{4.1}$$

$$Accuracyintermsofconfusionmatrix = \frac{(TP + TN)}{(TP + FN + FP + TN)} \tag{4.2}$$

Precision: Precision is defined as the proportion of instances classified as positive that are really positive.

$$Precision = \frac{TP}{(TP + FP)} \tag{4.3}$$

Recall:

Recall is defined as the proportion of positive instances that are correctly classified as positive.

$$Recall(Classifier) = \frac{TP}{(TP + FN)} \tag{4.4}$$

F1 Score: This measure is approximately the average of the two(precision and Recall) when they are close. The two measures are sometimes used together in the F1 Score (or f-measure) to provide a single measurement for a system. F1 score is harmonic mean of precision and recall

$$F - Measure = \frac{2(PrecisionxRecall)}{(Precision + Recall)} \tag{4.5}$$

4.3.2 Classification with Support Vector Machines

Support vector machines are binary classifiers that assign one of two labels to a test point as which side of the decision hyperplane g(x) = 0 the points lies on. However, many classification tasks involve more than two labels. In these cases we modify the SVM objective function in order to obtain a true multiclass classifierY. Lee et al. (2004); Crammer & Singer (2001); Cortes & Vapnik (1995), but as Hsu & Lin (2002) writes such solutions typically yield more complex optimisation problems and can take a very time for learning . A simple approach is to train a number of

standard SVM classifiers on binary subtasks and integrate their predictions on each test example to give a multiclass prediction. There are two approaches to do it. One against all and one against one. for ova one binary SVM is learned for each of the K classes. and ovo, a binary SVM is trained for each pair of labels. Geberally ovo approach is used for multiclass classification. There are different kernel functions used in SVM classifier Linear, Polynomial, Gaussian, Radial Basis Function (RBF), and Sigmoid. These functions determine the smoothness and efficiency of class separation. In SVM classifier, for hyperparameter tuning there are two methods: GridSearch and Random search. We have used GridSearch in our experiment to improve the accuracy of the result.

4.3.3 Classification with Random Forest Classifier

Random Forest is a very famous supervised machine learning algorithm used for classification and regression both problems. It is based on the concept of ensemble learning i.e to combine multiple classifiers into one for a complex problem and improve the model performance. It is an ensemble of several decision trees into one. It takes prediction of many random decision trees and calculates the average of the results and predicts the final output for classification. It is effective for multi class classification. More trees in the training data for learning better performance of the model. Random Forest works in two-phase: 1. to create the random forest by combining N decision tree 2. to make predictions for each tree created in the first phase.

4.3.4 Classification with BERT Classifier

BERT stands for Bidirectional Encoder Representations from Transformers. Recently BERT has been proved to be effective for many NLP tasks especially for Text

classification. BERT uses a Transformer model that learns contextual relations between words in a sentence or a corpus of texts. A transformer model includes 2 separate methods: an encoder that reads the text input and a decoder that generates a prediction for any given task. BERT uses only the encoder as its goal is to generate a language model. It is a pre-trained deep bidirectional representation from the unlabeled text by considering the both left and right context of the text. BERT uses two methods for training the language model : MLM (Masked LM) and NSP (Next Sentence Prediction) For classification method one uses Pretrained BERT model embeddings. First the input is supplied as token forms like CLS for classification tasks. BERT for learning takes a sequence of words, as input. The Self-attention layer is applied to every layer and the result is passed through a feed-forward network and then to the next encoder. After preprocessing of text the text is provided to BERt and BERT output layer encodes the language model and then classifies the sentences. For embedding based classification , the embedding layer output from BERT is applied to as input in the classification layer. The classifier can be a neural network classifier or a BERT classifier itself.

4.4 Word Embeddings

Recent advancements in NLP have used word embedding to represent the words and have proven to be successful to the extent that many NLP applications have used this for text classification to entity extraction , machine translation and semantic analysis ;sentence similarity or semantic relations extraction. Word embedding is based on the distributional semantic theory of linguistics and therefore used over the statistical analysis based representation on the large corpus. The word embedding can be obtained automatically from a large amount of unlabelled text, thus no need for manual annotation of the corpus.

The word embedding is based on a distributional semantic hypothesis. The hypothesis has its roots in theoretical linguistics to represent the words and meaning.Harris (1954); Firth (1957); Taylor (2003) Wittgenstein (1953). The hypothesis claims that the word has a similar meaning in a similar context. The linguistics items which have similar distribution tend to occur in similar context and have similar meanings. The model based on the hypothesis is used to represent the semantic similarity between the words. Distributional semantic models use Vector Space models and distributional hypotheses of lexical semantics. Vector space model is used to represent words or text in a document using a vector or identifier. DSMs have one vector per word representing the word based on its occurrence of other words in the same context. Word Embedding is learned using a real valued vector representation for a predefined fixed size of vocabulary from a corpus. The word vector is represented by counting the number of times the word co occurs with other words in the context.

Several earlier studies demonstrated that word embeddings not only capture word similarities but also some types of relational similarities between pairs of words Mikolov, Yih, & Zweig (2013); Levy & Goldberg (2014); Mikolov, Yih, & Zweig (2013). Furthermore, a number of recent studies have also used word embeddings as input representation to their neural models for noun–noun compound interpretation in particular Dima & Hinrichs (2015); Ponkiya et al. (2020); Dao et al. (2021); Shwartz (2019); Shwartz & Dagan (2019).

The very famous example to understand the word embeddings for capturing the syntactic and semantic regularities is given as: that subtracting the vector of the word man from the vector of king and adding the vector of woman results in a vector whose nearest neighbor is the vector of the word queen ; that is, if **wi** is the vector for the word **i**, the observation translates to:

$$\mathbf{wking} - \mathbf{wman} + \mathbf{wwoman} = \mathbf{wqueen}$$

Word embeddings based on a large trained corpus can provide a good foundation for the semantic interpretation of some complex NLP tasks. Compound nouns are usually formed from existing nouns in the language. However, the meaning of the compounds is often not the combinatorial function of the meaning of their constituents. It is constrained by the tendency of a word to combine with other words. This combinatory potential of the words is captured in a word embedding model. Thus, word embedding with machine learning classifiers appears suitable for experimenting with semantic relation classification tasks.

Word embeddings are of different types: Word2Vec, BERT embeddings, Glove embedding based on the learning algorithms and parameters. One can use existing pre-trained word embedding and fine tune it with their tasks or train their own embedding using their corpus from scratch. We have used pre-trained word embedding and fine tuned it based on our task.

4.4.1 Word2Vec Embedding

A word embedding W : D → Rn is a function that assigns each word from the embedding dictionary D an n-dimensional, real-valued vector Dima (2016). The words in the dictionary D are embedded in a high-dimensional vector spaceSorokin et al. (2015) . Word2vec embedding is a statistical method to learn standalone word embedding based on text corpus. The word2vec does not consider the contextual variation meaning of the words. For example the word2vec for the word bank would be the same for any meaning; bank of the river or bank as financial institution. While the contextual embedding like BERT will produce two different vectors for the word bank depending on the meaning and context of the words. Word2Vec can capture the context of a word and the relation between words, and the representation of syntactically and semantically similar words. Word Embeddings result from training language models on large amounts of unlabeled text corpus using various

learning algorithms. The embedding can be obtained using two methods: Continuous bag of words method or continuous skip gram model. The input for word2vec is a text corpus and the output is the vector representation of words, the feature vector.

4.4.2 BERT Embedding

BERTDevlin et al. (2018) is a language model trained on a large unlabelled corpus to represent deep contextualized representations of words. The BERT model has given state-of-the-art results in many NLP tasks and has proven to be effective in extracting contextual features and word features from sentences for many semantic tasks. BERT embedding has been used in word sense disambiguation tasks Peters et al. (1802)to represent the word features. The BERT language model has been used in the BBC Hindi text corpus to classify the news text into different categories and perform the NER task. BERT architecture uses transformers, encoders, and decoder models. A transformer is an encoder-decoder network that uses self-attention on the encoder side and attention on the decoder. BERT has an input level of text and then hidden layers, which give embedding of words as output, and this output is fed into the classifier as features for the classification task.

4.5 Compound Interpretation Dataset

In this section, we introduce the dataset which we will use in our machine learning experiments for compound interpretation in the chapter. The dataset prepared is based on our dataset which we have discussed in chapter three in detail. The dataset contains 1500 general domain compound nouns extracted from IIT bombay multiword expression corpus annotated with 20 semantic relations.

The previous chapter presented a detailed account of the preparation and annotation of Hindi Compound nouns using different semantic relations and backed up the data with calculating the inter annotator agreement.

For automatic interpretation of Hindi Compound nouns using machine learning we have created the 1500 compound noun dataset. The dataset is created from the list of 12000 multiword expression lists developed at IIT Bombay. The following procedure is followed for the preparation of data. We used the 600 general domain annotated compound noun data presented in the previous chapter and then using 20 semantic relations sets we annotated more 900 compound noun data from the multiword list. Thus prepared 1500 compound noun data annotated with 20 semantic relations. We required all the compound nouns having two noun sequences.

4.5.1 Computation of semantic relations

Humans understand the meaning of a word using their world knowledge and commonsense. Human minds have semantic features of the words. For developing a model for semantic analysis we need some resources which can provide semantic information, linguistic information and world knowledge.

4.5.1.1 Computational model for semantic relations identification

The computational approaches used in Hindi compound noun interpretation using machine learning comprises machine learning techniques to treat compound interpretation as a classification task using a standard set of semantic relations as the prediction labels; Noun features, Hindi WordNet features and word embedding output as the input features to represent the constituents and machine learning classifiers to classify the relations.

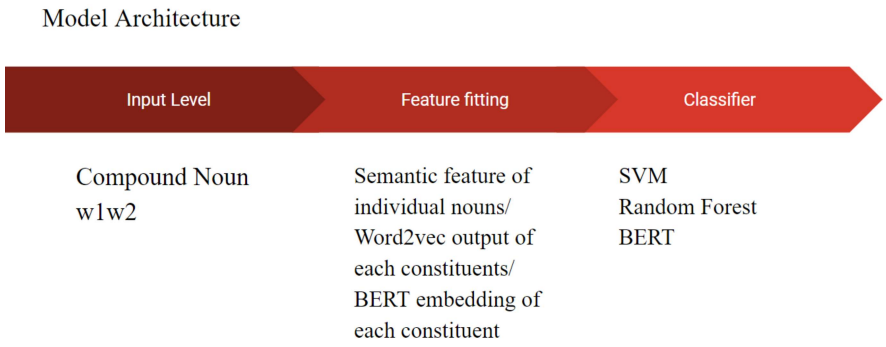

FIGURE 4.1: The computational model for compound noun interpretation

We performed several experiments using different features with different classifiers and evaluated the model output using F1 score.

Model Architecture: The computational model is shown in the figure 4.1.

4.5.2 Computation of Lexicalized, Name and Dvandva relation

We excluded the Lexicalized, Names and Dvandva relations using a python program from our final 1500 compound nouns data.

As described in chapter 3, The lexicalized relations are the relations in a compound noun whose meaning can not be understood by the constituents meaning, Bahurivi type compounds. We found that all the compound nouns tagged as lexicalized relations are included in Hindi WordNet. Therefore, using a python program we classify the lexicalized relations for the compounds which are found in the WordNet. The limitation of this approach is that if a compound which is actually lexicalized occurs in our data which is not found in wordNet our model would provide null relation.

Dvanda relation has and relations between the constituents, our data does not contain much of the compound of such type and most of them are written with hyphen in

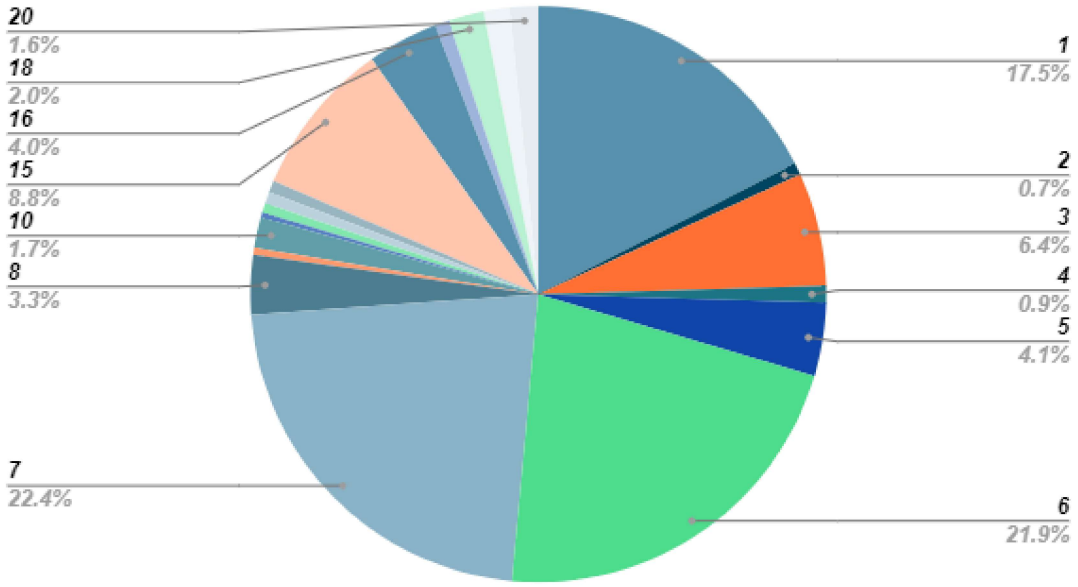

FIGURE 4.2: Distribution of compound nouns into different semantic relations

between them. Our model would mark any compound noun having hyphen between them as dvandva. This rule is developed based on our dataset. This is generally not acceptable in literature. As copulative compounds are a very important feature in Indian languages. This relation needs its individual deep study alone. There are different types of copulative compound nouns.

In our data there were many compounds which are proper names, like *sambalpura jilaa, durgaa mandira, khajuraho mandira.* We excluded these types of relations from our dataset using a python program making the list of these proper names.

From the remaining 17 relations 60% of the Compound nouns have PURPOSE, MODIFIER and TOPIC relations. And the other 40 % included all remaining relations.

The distribution of frequency of semantic relations is given in the figure 4.2.

Relation	Paraphrase	Example
Purpose (for):	N2forN1	/kar indastri/ 'car industry'.
Modifier relation	N1 modifier of N2	/bhartiya dhaneS/'Indian hornbill'
Topic	N1 related to N2	/namak kanun/ 'salt law'
Other	N1 used for N2	/bas yatra/ 'Bus travel'

Table 4.2: Semantic Relation

4.5.3 The compound noun dataset used with the classifiers

Machine Learning requires a good amount of data to train the model. The relations having small amounts of data will increase the model complexities in identifying features and hinder the learning process. The result will be the underfitting of the model for some relations and overfitting for some. This will affect the model performance and output accuracy. The dataset used for the experiments consists of 1500 Hindi Compound Nouns annotated with 20 semantic relations. We classified the entire set of data into four categories. Purpose, Modifier, Topic and Other. The Purpose, Modifier, and Topic relations cover the majority of the semantic relations of the Hindi compound nouns. Table 4.2 represents these relations with examples, and figure 4.3 shows the relative frequency of each relationship within the dataset. The highest frequency relation in our dataset is from the Others relation. This relation consists of a set of relations having significantly less frequency in our dataset. Three major relations found in our dataset are Purpose, Modifier, and Topic. Purpose relation presents the relationship between constituents as N2 is used for N1, E.g., *rasoi ghara* cooking room, 'Kitchen' where ghar 'room' used for rasoi 'cooking'. Similarly, Modifier and Topic relation can be represented as N2 modifies N1 and N1 related to N2/N1 is the topic of N2, respectively. The other relation consists of minor relations like the Dvanda relation, where both N1 and N2 are independent concepts and equally important in the compound word, Instrument relation, where N2 is an instrument for N1. Then we divided our data into two sets: training set and testing set. The semantic relation example is given in the table 4.2

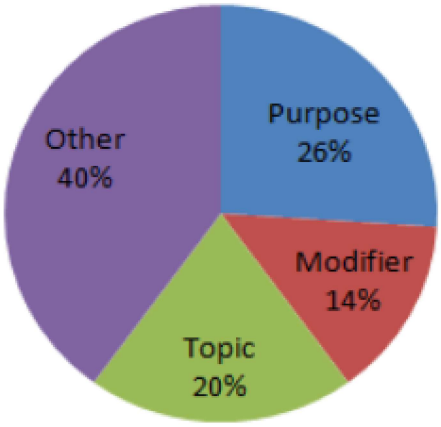

FIGURE 4.3: Distribution of frequent semantic relations with their frequency

The distribution of frequency of semantic relations is given in the figure 4.3.

4.6 Pre Processing of Data

For use in machine learning algorithms the preprocessing of text is required. For preprocessing first we removed all the inflections found in the compound nouns. Since Hindi Compound nouns mostly come with inflections marking number, and gender in running text. The multiword expression list developed at IIT Bombay is obtained on a running text corpus; it needs preprocessing for further use in the model. Some of the compounds have hyphens between them, some were written as single word and some as double wod. After the preprocessing of the text we curated a 1500 compound noun dataset containing a noun-noun sequence written as two words with a space between them.

4.7 The Classification of semantic relations using Noun features and Machine Learning Classifier

4.7.1 The Classification of semantic relations using Noun features

This section demonstrates the model for semantic relation classification between the constituents using individual noun features as input feature for compound nouns and then evaluating the model with F1 score.

The approaches used in this work comprises machine learning techniques to treat compound interpretation as a classification task using a standard set of semantic relations as the prediction labels and machine learning classifiers to classify the relations.

We have used SVM and Random Forest classifiers for classification purposes. SVM classifiers work better with binary classification and recent SOTA works on compound noun interpretation used SVM classifiers in their classification model. We chose Random Forest to compare our classification accuracy with other models. Random Forest is said to be the best for multiclass classification.

Our model architecture is based on the hypothesis that the semantic category of individual constituents influence the semantic relations. For example if the modifier noun N1 is a noun derived from adjectives the resulting compound noun will be classified as modifier relations. A verbal noun head takes a proper noun as a modifier. Like saikila yatraa yatra will take saikil an inanimate noun as an object. Some types of noun-noun combinations can give the same relations. Therefore we used individual noun features as input features to classify the semantic relations.

4.7.2 Noun Features

We classified the nouns of the compounds for attributing features to the ML system. Following types of nouns were used nouns for classification of the most frequent semantic relations

1. Activity noun (ActN) 2. Human noun (HN) 3. Common noun (CN) 4. Organization noun (OgN) 5. Place noun (PlN) 6. Domain noun (DomN) 7. Program noun (ProgN) 8. Abstract noun (AbsN) 9. System noun (SysN) 10. Animate noun (AnN)

For the Modifier relation, we also classified some adjectives which come before the nouns in the compound. These are the following: 1. Time Adjective (TA) 2. Qualitative Adjective (QA) 3. Adjective from Proper Noun (PrNA) 4. Adjective from Common Noun (CA)

4.7.3 Experiments

For classification of the Nominal Compound using the SVM and Random Forest machine learning approach, we have used the following feature set. The features used are as follows: Type of noun of the first word of the Nominal Compound
The input is provided into SVM and Random Forest classifier and then we trained the model on training data and evaluated the model based on our test set. Type of noun of the second word of the Nominal Compound The next we divided our data into training(80%) and testing(20%) part.

We performed several experiments using this model: First we experimented for multiclass classification using SVM and then Random Forest classifier. Then we experimented with binary class classification for most frequent relations: PURPOSE, MODIFIER TOPIC and OTHER. We performed the experiments using two methods.

MODEL	F1 Score
SVM	35.3
Random Fores	48

Table 4.3: Result from SVM and Random Forest for multi class classifier

Classifier	Predicted Class		Actual Class
	Affirmative	Negative	
SVM(purpose)	12	20	Affirmative
	13	130	Negative
RF(Purpose)	8	30	Affirmative
	132	15	Negative
SVM(Modifier)	1	8	Affirmative
	159	7	Negative
RF(Modifier)	1	15	Affirmative
	157	2	Negative
SVM(Topic)	10	38	Affirmative
	116	11	Negative
RF(Topic)	25	33	Affirmative
	101	16	Negative

Table 4.4: Confusion matrix of all three relations classification results

First method used a one against all approach i.e. all data is classified into two relations one positive and one negative for every experiment and for each semantic relations. Positive being purpose, modifier and topic and negative being non-purpose,non modifier and non topic.

Second method involves one against one approach i.e for positive we used purpose and for negative we used Modifier. Similarly, using this approach we performed the experiments with other relations combinations and then evaluated the model performance.

We also checked the best hyper parameter for SVM using GridSearch CV method.

4.7.4 Results and Discussion

Total number of semantic relations in our data:

1. No of valid Purpose relations -164

2. No. of valid topic relations—342

3. No. of valid Modifier relations—109

We observed from the results that both the models could not perform better for multiclass classification.The previous experiment using SVM and Random forest for binary class classification yields more negative predictions than positive predictionsDwivedi & Ghosh (2022). Due to the imbalanced data. We had around 30% of purpose relations and 70% of other remaining relations. In this experiment Random Forest performs better than SVM for predicting negative classes more accurately.The confusion matrix in the table **??** for each three classifiers in one against all approach depicts that the negative prediction score was better than the positive score. The reason for this is that for the negative category we had more examples in training than for positive one.

To improve the accuracy of the model, the performance we used a similar amount of data for experiments for each relationship and trained the model. The result using one against one was improved from our earlier experiment with more positive predictions. In this experiment our SVM model performs slightly better than the Random Forest classifier. The best model was SVM with the RBF kernel in this experiment. as the graph illustrates in the figure 4.4 SVM and Random Forest both gave same accuracy for modifier relation . Thus, we showed that the category which has more examples in training data is predicted more accurately than the one which has less example instances. This is certainly the assumption behind machine learning algorithms to perform better.

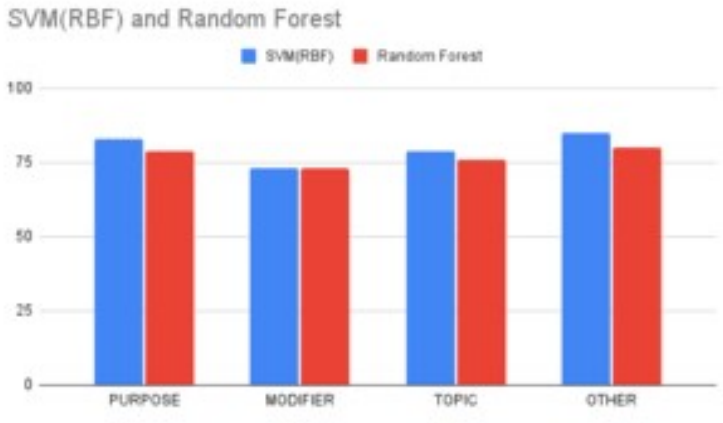

FIGURE 4.4: Binary Class classification result comparison using SVM and Random Forest(F1 score)

Overall we can say that SVM and Random Forest both are good classifiers for classifying the semantic relations. For Random Forest to work better there should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result. The predictions from each tree must have very low correlations. Since our data was sparse and the features were more related. The Random Forest could not perform better for binary class classification.While for multi class classification Random Forest performs better with SVM classifier.

Our result also demonstrates that only semantic category of individual noun is not sufficient for model to predict the relations. Most of the nouns overlap with each other into their semantic category, this can confuse the model. We need to use more features for correct predictions and for all the semantic relations predictions.

Our approach predicts only four type of semantic relations from the 20 semantic relations using machine learning.

Therefore, We conclude that if the data is balanced and with sufficient features are available the machine learning model is used for automatic interpretation of compound nouns.

4.8 The Classification of semantic relations using Word Embedding and SVM and BERT Classifier

This section demonstrates the model for semantic relation classification between the constituents using word embeddings as input feature for compound nouns and then evaluating the model with F1 score.

The approach used in this work treats Compound Noun Interpretation as a classification task using semantic relations as prediction labels and word embeddings as features with SVM and BERT classifiers. We performed two experiments: one with word2vec embedding as input features and SVM classifier other with BERT embedding as input features and BERT classifier.

As we explained in the previous section, for machine learning experiments we need balanced data therefore we prepared the balanced data for further use in semantic relation classification tasks. The dataset used for this experiment is the same as the previous dataset. The compound nouns with 1200 compound nouns are classified into four major relations; PURPOSE, MODIFIER,TOPIC and Others.

The objectives of the experiments were twofold. 1. To classify the semantic relations between the two-word compound nouns of Hindi using a scheme of multi-class comprising Purpose, Topic,Modifier, and other. 2. To test the performance of the BERT classifier and Compare it with the SVM classifier for a difficult semantic interpretation task.

For compound noun interpretation using Word Embeddings,we assume that the two compound nouns having the constituents a,b and c,d and the semantic relation between these two compound nouns is same then the semantic similarity between the constituent a and c or b and d will be the same for the relation to be the same.

For training the model we will look for the compound nouns c, d with the highest similarity compound a, b and thus gives the relation of a,b to c,d. For example in the compound nouns *saikila yatraa (bicycle ride) and rela yatraa (rail journey)* ; the words *saikila and rela* would be similar since yatraa is same in both the compound nouns. The representation of the compound nouns will be:

saikila yatraa :: rela yatraa => saikila: rela:: yatraa: yatraa => Instrument relation

The vector representation output from the word2vec and BERT embedding of two similar words will have similar representation and thus will give the same semantic relation for the test. We have used the same dataset described in this chapter for experiment.

4.8.1 The Experiment 1: Word2Vec Embeddings and SVM classifier

We used word2vec embedding as a feature set for our model for the experiment. The Compound Noun Classification experiment uses a Hindi word2vec embedding obtained from IIT Bombay Hindi monolingual corpus using Gensim word2vec. Word2Vec embeddings Mikolov, Chen, et al. (2013) of dimensions 50, 100 for both skip-gram and CBOW architectures are created using the Gensim library Řehřek et al. (2011) implementation of Word2Vec. The output of word2vec embedding for individual constituent work as a feature. The dataset is then partitioned into a training and a testing set. We used 80% data for training and 20% for testing, 1200 and 300 compound nouns, respectively. SVM classifiers are generally used for binary classifiers, so we break down the multi-class classifier into multiple binary classifiers. We developed two different classifiers using two different kernel functions; Polynomial and RBF. We trained the model using the word2vec features and relation labels and then calculated the performance of our model using the F1 score and accuracy.

The classification is done for four frequent semantic relations classes(PURPOSE, MODIFIER, TOPIC and OTHERS) as discussed in the dataset section.

4.8.2 The Experiment 2: BERT Embeddings and BERT Classifier

For another experiment, We used a pre-trained BERT model for the Hindi Text classification task trained on BBC Hindi news dataset and then fine tuned that model for our task. We divided our dataset into testing(30%) and training(70%) dataset. The embedding layer is obtained by using BERT and then we used mBERT classifier to classify the different semantic relations. mBERT (Multilingual bert) is a BERT based language model which has sentence representation of 104 languages around the world. The output of the embedding layer is an output vector of each token and then we feed this output vector to the classification layer and train the model. We then evaluated the model based on our test data using f1 score. We also checked individual semantic relations prediction. The classification is done for four frequent semantic relations classes(PURPOSE, MODIFIER, TOPIC and OTHERS) as discussed in the dataset section.

4.8.3 Result obtained From SVM classifier

This section discusses the result of the experiment conducted using word2vec embedding (section 4.6) and an SVM classifier presented in the previous section. The SVM model is trained using splits in data into training (80) and testing set (20) for multiclass classification. We have used two kernels, Polynomial and RBF, for the classification. We have used accuracy and F1 score to measure the model's performance. The result obtained from the experiment is presented in Table.

MODEL	F1 Score	Accuracy
SVM(RBF)	47.94	63.46
SVM(Polynomial)	46.54	61.26

Table 4.5: Result for SVM classifier

MODEL	ACCURACY	EVAL LOSS
BERT	0.42	1.22

Table 4.6: F1 score for BERT classifier

The table shows the result in terms of the F1 score and accuracy of the model. The accuracy score shows the percentage of the true positive and true negative to all given data points. The f1 score calculates the harmonic mean between precision and recall, and both depend on the false positive and false negative. The result shows that SVM with Polynomial kernel function outperforms SVM with RBF kernel function.

4.8.4 Result from BERT classifier

This section describes the result obtained from the experiment done using the BERT classifier to classify the relations and compare it with other existing works. The semantic relation classification for Hindi Compound nouns using BERT classifier gives 48% accuracy in terms of F1 score with evaluation loss 1.22. The f1 score is significantly good but the evaluation loss we got is not as par standard. The high evaluation loss means our model is over fitting. The reason for Over-fitting is due to the pretrained model which we have used is based on BBC Hindi news corpus that is used for different tasks as compared to ours and the embedding output of this corpus is based on sentences but we are using compound nouns. So the model has taken more features than required to classify the task. The result of our model is given in the table 4.6.

The accuracy of individual semantic relations are presented in table 3. The accuracy of others relations is highest in comparison to remaining relations due to the high

Semantic Relation	Total Frequency	Correct Predicted Frequency	Accuracy
Purpose	80	38	47.5
Modifier	95	45	46.3
Topic	67	24	35.8
Others	125	82	65.6

FIGURE 4.5: Result from BERT classifier for each semantic relation

frequency of this relation in data. As we can observe from table 3 from a total of 80 instances of purpose relation present in testing data our model could correctly predict 38 instances of compound nouns out of 80, giving 47.5% accuracy. Similarly Modifier, Topic and Others results are provided in Table4.5 .

4.9 Results and discussion

The previous work on the same data set using only SVM classifiers had an accuracy of 35% DWIVEDI & GHOSH (2022), but adding Word2Vec has significantly increased the accuracy of the interpretation to 63%. Analyzing the result from Table one, two and three , we show that the BERT outperforms the SVM classifier. SOTA result in Compound Noun Interpretation has been obtained by Tratz and Hovy, the experiment got 79% result using the machine learning models on English Dataset. The result of Tratz and Hovy is best till now with machine learning models and features. The SOTA result using embeddings has been obtained by Dima and Henrick (77%) and Ponkia et al (66%) on the English Dataset. All the SOTA results are obtained on English Datasets and Using English resources Viz, Wordnet,Word embeddings and Language models. Our result achieved 63% accuracy highest so far using Word embedding and SVM classifier on a Hindi Dataset. The reason for low accuracy is due to the less resources available for Hindi. BERT language model for Hindi is also not exhaustive and is trained only on a limited size of corpus. Word2vec output uses the features of individual nouns while BERT embedding also takes consideration of

Author	Model	Result
Tratz & Hovy	English dataset with ML algorithm	79%
Dwivedi & Ghosh	Hindi dataset with ML algorithm	35%
Dima & Henrich	English dataset with Embeddings	77%
Ponkia et al.	English dataset with BERT Embedding	66%
Our Model	Hindi dataset with Word2Vec and SVM classifier	63%
Our Model	Hindi dataset with BERT embedding and BERT classifier	48%

TABLE 4.7: Comparison of SOTA results to our results

context for giving the vector representation. Our semantic relation classification is based on the contextual interpretation of the compound; the SVM classifier with Word2vec output performs lower than BERT. BERT embedding is trained on BBC Hindi News corpus and our dataset is developed from the MWE expression list. Therefore, the discrepancies in the training pretrained embedding on different data and fine tuning the BERT model on different data yields lower results as compared to the general SOTA. From the experiments and the results based on our dataset we generalize that embedding captures semantic information in depth and thus help in achieving better accuracy.

Semantic analysis needs strong semantically rich resource with all the information encoded. Word sense also affects the performance of the model as a word has one meaning in a context and different meaning in other context.Our model did not take this into consideration therefore, less accuracy of prediction.

Comparison of SOTA results to our results is given in the table 4.7.

4.10 Conclusion

This chapter discusses and compares the three experiments we have done with different Machine Learning models.

We compare all the existing SOTA models for compound noun interpretation for English language and observed that the best result using machine learning is Rosario & Hearst (2001) with 60% accuracy for domain specific dataset and Tratz & Hovy (2010) with 79% accuracy for general domain corpus using maximum entropy classifier. Using different word embeddings Dima and Henrich(2015) got 77% accuracy on Tratz and Hovy general domain dataset. Rallapalli & Paul (2012) work which used Ontology and rule based approach on the Hindi Compound noun interpretation using a dictionary. The dictionary is used to translate the Hindi compound nouns into English compound nouns and then Ontology is used to classify the compound nouns. She got 32% accuracy for Hindi Compound Nouns. In our classification model for multiclass classification random forest and individual noun feature got best performance. For binary class classification SVM and Individual noun feature outperform all the classifiers. Our model predicted only four major classes of relations found in the data. We need to work more for other classes predictions.

The Tratz and Hovy work on compound noun interpretation used four different linguistics resources and a compound noun dataset of 17200 compounds. This work is the SOTA in compound noun interpretation task using machine learning. It proves the need of linguistics resources for compound noun interpretation task.

The chapter concludes that our model needs to be combined with some large database for training as well as a good lexical semantic knowledge base for better performance and result.

In the next chapter, we will discuss one such model of lexical knowledge representation which can be used in future works of computational semantics and information retrieval.