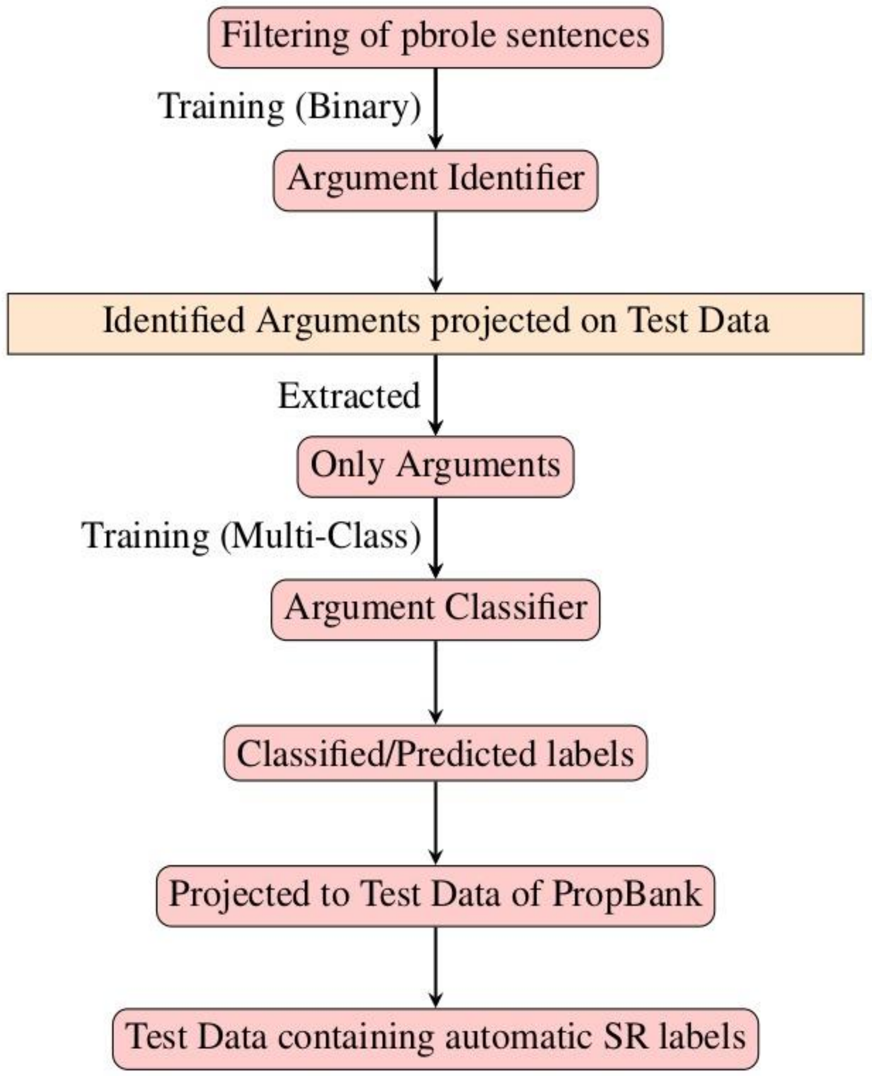
Overview



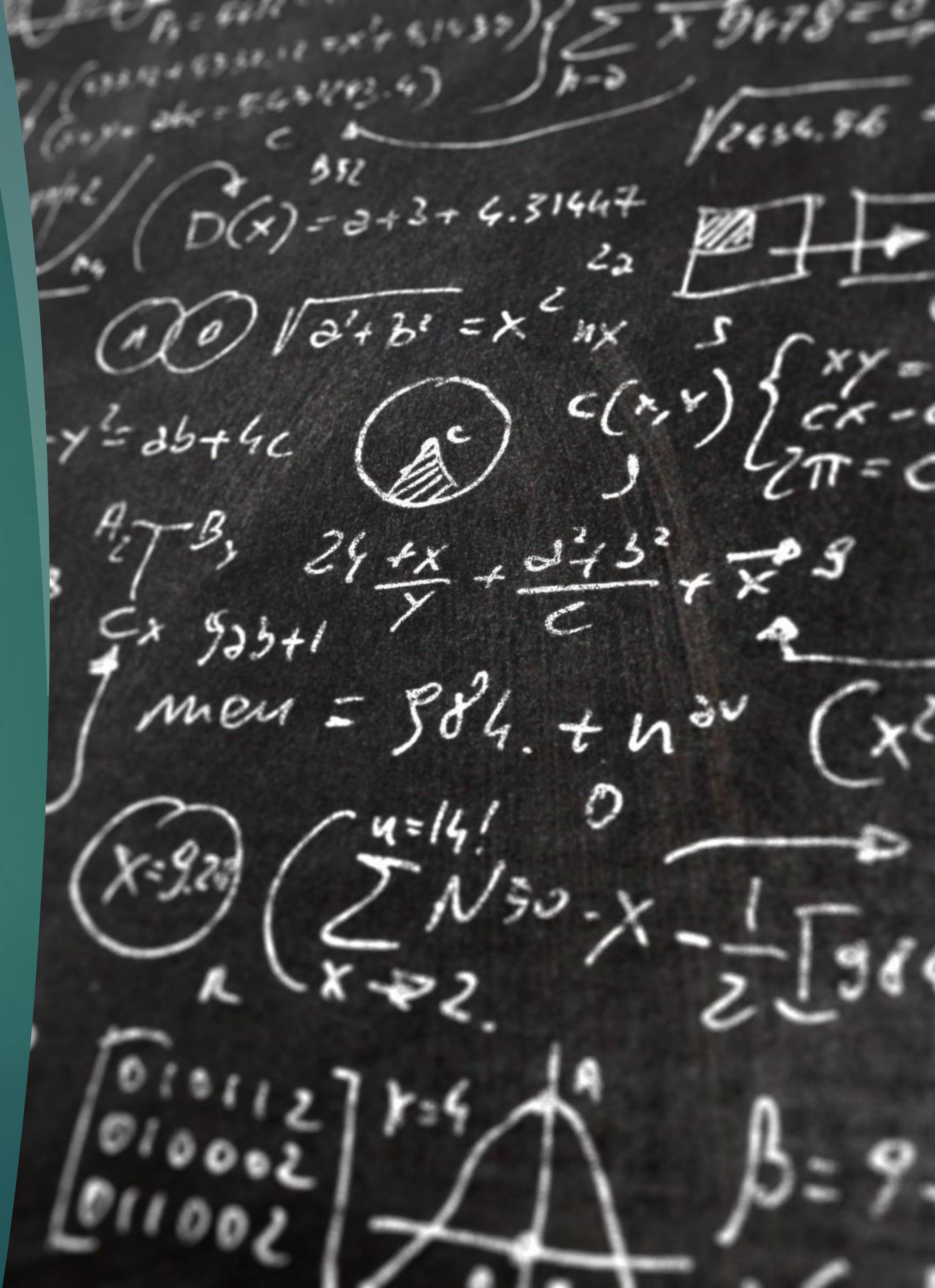
Filtering of pbrole sentences
Training (Binary)
Argument Identifier
Identified Arguments projected on Test Data
Extracted
Only Arguments
Training (Multi-Class)
Argument Classifier
Classified/Predicted labels
Projected to Test Data of PropBank
Test Data containing automatic SR labels

Figure 1: **SRL Architecture**

Abstract

Given an input sentence and a predicate/verb, the system first identifies the arguments pertaining to that verb and then classifies it into one of the semantic labels which can either be a DOER, THEME, LOCATIVE, CAUSE, PURPOSE etc.

Semantic Parsing is essentially the research investigation of identifying WHO did WHAT to WHOM, WHERE, HOW, WHY and WHEN etc.



Dataset, Embeddings and Models

Dataset

1. Hindi treebank and Propbank
2. 1600 sentences used

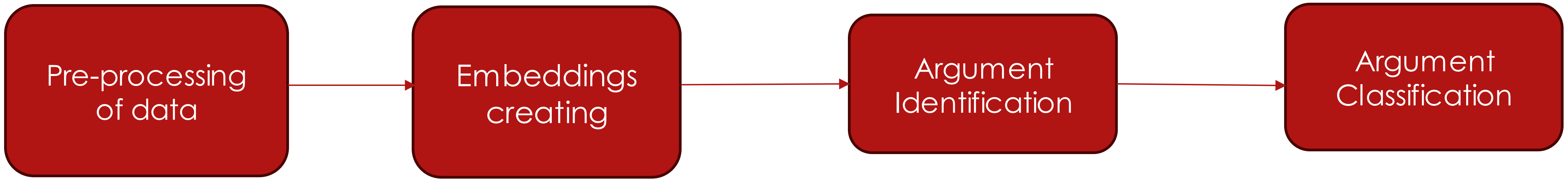
Embeddings

1. Word2Vec
2. FastText

Models

1. Logistic Regression
2. SVC(Support Vector Machine)

Flowchart


Pre-processing of data
Embeddings creating
Argument Identification
Argument Classification

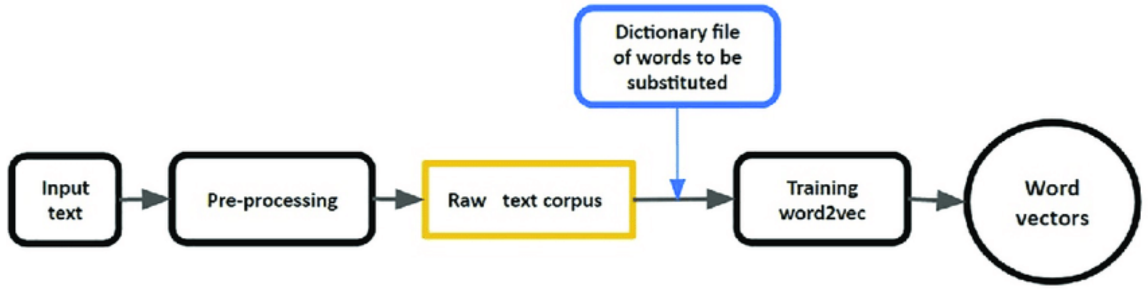
Argument Identification

- Utilized lexical (embeddings, types, lemma), syntactic (POS tags, dependencies), and contextual features for word characterization.
- We used a labeled dataset where each word in a sentence was labeled as either an argument or non-argument
- Features were extracted for each word in the sentence.
- Trained the model on extracted features.
- Evaluate precision, recall, and F1-score metrics, for the task of argument identification

Argument Classification

- Use a model to assign semantic labels (e.g., Agent, Theme, Location) to identified arguments based on their role in the sentence.
- Engineering features to capture lexical, syntactic, and contextual information for representing each identified argument effectively.
- Training the model using a labeled dataset where each argument is associated with a specific Propbank label representing its semantic role.
- Evaluating the classifier's performance using metrics such as accuracy, precision, recall, and F1-score to assess its effectiveness in assigning semantic roles to arguments.

Word2Vec



Semantic Context: Word2Vec embeddings encode semantic meaning by representing words as dense vectors in a continuous space.

NLP Efficiency: Word2Vec embeddings enhance NLP tasks with efficient, context-aware word representations that preserve semantic relationships.

Pros & Cons

Pros:

Efficiently captures semantic relationships between words in a continuous vector space.

Provides dense and context-aware word representations beneficial for downstream NLP tasks.

Cons:

Requires large amounts of training data to learn accurate embeddings.

May struggle with out-of-vocabulary words and rare terms not encountered during training.

FastText

- **Subword Information**: FastText embeddings capture morphological information by considering character n-grams, enabling it to handle out-of-vocabulary words and morphologically rich languages effectively.
- **Enhanced Generalization**: FastText improves generalization by leveraging subword representations to encode semantic and syntactic information, benefiting tasks like text classification and named entity recognition.
- **Language Flexibility**: FastText is adept at handling diverse linguistic patterns, making it suitable for languages with complex morphology where words share common prefixes or suffixes.
- **Efficient Training**: Despite incorporating subword information, FastText remains computationally efficient, making it scalable to large datasets and practical for real-world NLP applications.

Pros and Cons

Pros:

Sub-word Information: Fast-Text generates embeddings based on character n-grams, allowing it to capture morphological variations and handle out-of-vocabulary words more effectively compared to traditional word embeddings.

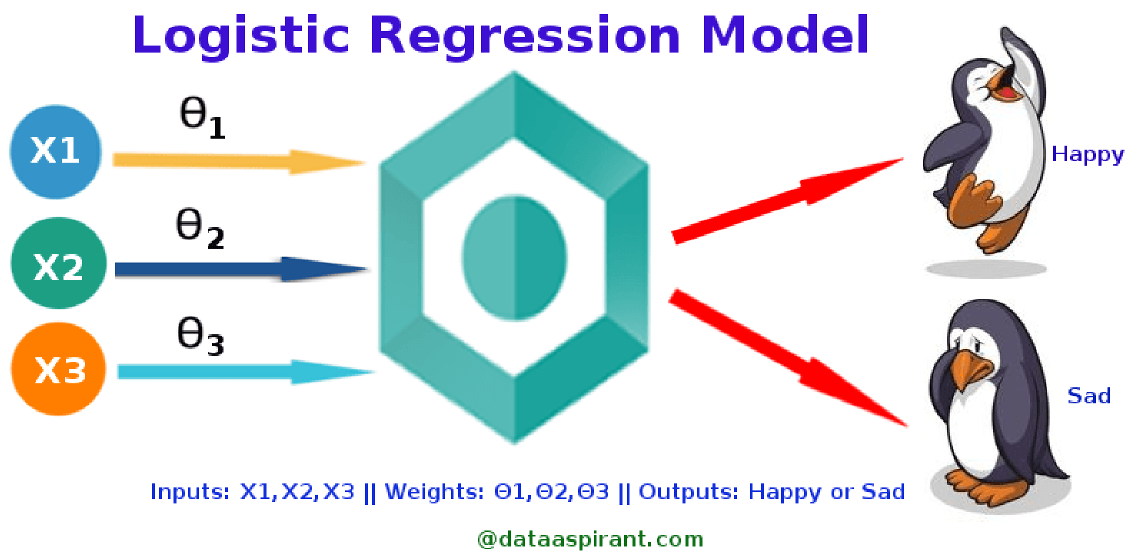
Language Flexibility: FastText is well-suited for morphologically rich languages where words share common prefixes or suffixes. It can capture finer linguistic nuances and improve the performance of NLP tasks in these contexts.

Cons:

Increased Model Size: Including subword information can result in larger model sizes compared to traditional word embeddings like Word2Vec, which may require more memory and computational resources during training and inference.

Complexity and Training Time: Training FastText models with subword information can be more computationally intensive and time-consuming compared to simpler word-based embeddings, especially when dealing with large datasets or extensive vocabularies.

Logistic regression


Logistic Regression Model
X1
X2
X3
θ1
θ2
θ3
Happy
Sad
Inputs: X1,X2,X3 || Weights: Θ1,Θ2,Θ3 || Outputs: Happy or Sad
@dataaspirant.com

- **Binary Classification**:
  - Logistic regression is a popular statistical method used for binary classification tasks, where the goal is to predict a binary outcome (e.g., yes/no, true/false) based on input features.
- **Probabilistic Modeling**:
  - Logistic regression models the probability of a binary outcome using a logistic (sigmoid) function, which outputs values between 0 and 1, representing the likelihood of the positive class.

Pros and Cons

Pros:

Interpretability: Logistic regression coefficients provide insights into the relationship between input variables and the probability of the outcome, allowing for easy interpretation of model results.

Efficient and Fast: Logistic regression is computationally efficient and scales well to large datasets making it suitable for quick model training and inference in binary classification tasks.

Cons:

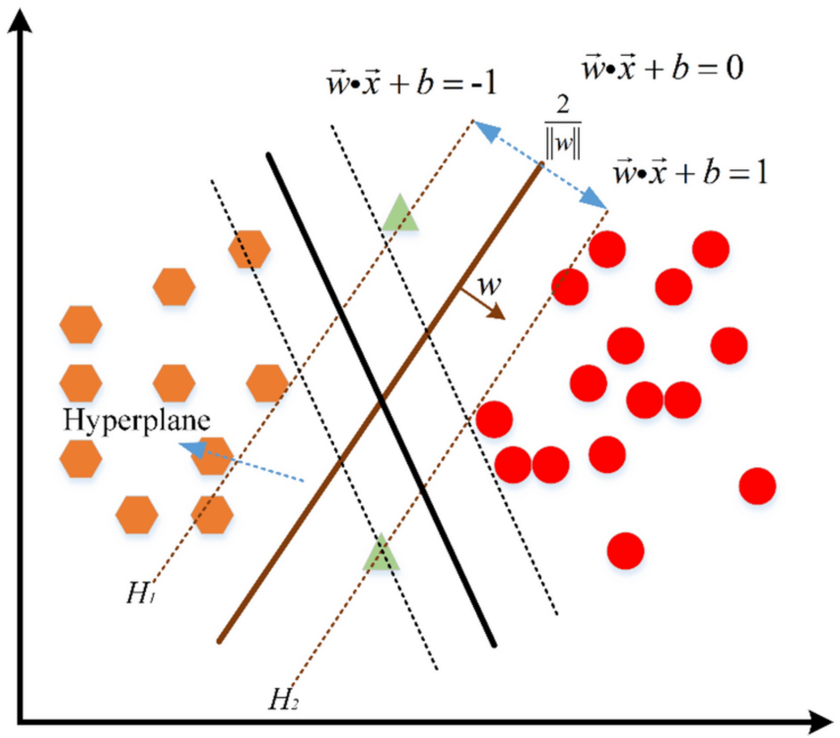
Linear Decision Boundary: Logistic regression assumes a linear decision boundary between classes, which may not capture complex relationships in the data effectively.

Limited to Linear Relationships: Logistic regression is limited to modeling linear relationships between input variables and the log-odds of the outcome, which can be restrictive for capturing non-linear patterns in the data.

SVC

Effective for Non-linear Data: SVC is a powerful algorithm capable of finding complex decision boundaries by mapping data into higher-dimensional spaces using the kernel trick, which helps capture non-linear relationships between features.

Margin Maximization: SVC aims to maximize the margin (distance) between the decision boundary and the closest data points (support vectors), leading to better generalization performance and robustness against overfitting.


$\vec{w}\bullet\vec{x}+b=-1$
$\vec{w}\bullet\vec{x}+b=0$
$\frac{2}{\|w\|}$
$\vec{w}\bullet\vec{x}+b=1$
w
Hyperplane
H_1
H_2

Pros and Cons

Pros:

- **Effective in High-Dimensional Spaces**: SVC performs well in high-dimensional spaces, making it suitable for tasks with many features, such as image classification and text classification.
- **Robust to Overfitting**: SVC maximizes the margin between classes, which helps in generalizing well to unseen data and reduces the risk of overfitting, especially when using appropriate regularization techniques.

Cons:

- **Computationally Intensive**: SVC can be computationally expensive, especially with large datasets, as it involves solving a quadratic programming problem to determine the optimal decision boundary.
- **Sensitive to Noise and Outliers**: SVC is sensitive to noisy data and outliers, as it tries to maximize the margin based on support vectors, which can lead to suboptimal performance if the dataset contains significant noise or outliers

Logistic regression with word2vec

Head with both Dependency

Identification

	precision	recall	f1-score	support
accuracy			0.83	4364
macro avg	0.83	0.84	0.83	4364
weighted avg	0.85	0.83	0.83	4364

Classification

	precision	recall	f1-score	support
accuracy			0.59	4364
macro avg	0.04	0.05	0.05	4364
weighted avg	0.41	0.59	0.48	4364

Logistic regression with FastText

Head with both Dependency

Identification

	precision	recall	f1-score	support
accuracy			0.82	4364
macro avg	0.81	0.82	0.81	4364
weighted avg	0.83	0.82	0.82	4364

Classification

	precision	recall	f1-score	support
accuracy			0.66	4364
macro avg	0.23	0.12	0.13	4364
weighted avg	0.60	0.66	0.61	4364

SVC with Word2Vec

Head with both Dependency

Identification

	precision	recall	f1-score	support
accuracy			0.83	4364
macro avg	0.82	0.84	0.83	4364
weighted avg	0.84	0.83	0.83	4364

Classification

	precision	recall	f1-score	support
accuracy			0.68	4364
macro avg	0.07	0.08	0.07	4364
weighted avg	0.54	0.68	0.59	4364

SVC with FastText

Head with both Dependency

Identification

	precision	recall	f1-score	support
accuracy			0.83	4364
macro avg	0.82	0.84	0.83	4364
weighted avg	0.84	0.83	0.83	4364

Classification

	precision	recall	f1-score	support
accuracy			0.70	4364
macro avg	0.19	0.12	0.12	4364
weighted avg	0.63	0.70	0.65	4364