CS - 6320
Natural Language Processing
Fall 2020

Course Project

Name	**NetID**
Rakesh Kumar Mahato	RKM190000
Nada Tade	NXT180027

A. Project Description

Design and implementation of models for extracting semantic relations between two named entities in a sentence. There are two named entities “e1” and “e2” in a sentence. A relation and its direction that holds between the two entities is provided. We had to design an algorithm to learn from the training examples and classify the entities among the following relations: Entity-Destination, Cause-Effect, Instrument-Agency, Content-Container, Component-Whole, Entity-Origin, Message-Topic, Product-Producer, Member-Collection, and Other.

B. Proposed solution

Collect the features from the input sentences and feed it to a machine learning model and use the model for prediction.

We studied few examples from the training dataset to understand the relation between the entities “e1” and “e2”. There are different types of cases that relate the two words.

Example 1:

Arcane Subtlety reduced the threat caused by Polymorph by 40% at max rank, though the <e1>threat</e1> caused by the <e2>spell</e2> is minimal.

Relation: Cause-Effect(e2,e1)

We see that the phrase “caused by” in between the two entities give a hint towards a Cause-Effect relation.

Example 2:

Many other <e1>dwarves</e1> also hailed from this infamous <e2>clan</e2>.

Relation: Entity-Origin(e1,e2).

We see that the phrase “hailed from” in between the two entities give a hint towards a Entity-Origin relation.

Example 3:

This book has transported <e1>readers</e1> into <e2>ancient times</e2>.

Relation: Entity-Destination(e1,e2)

We see that the phrase “into” in between the two entities give a hint towards a Entity-Destination relation.

C. Implementation Details

Writing this code required us to understand the basics of English grammar and sentence formation. We have implemented our code in Python Programming Language.

Libraries used:

- spacy
- nltk
- time
- sklearn
- verbnet
- wordnet
- csv

Step1: Feature extraction

1. Lexical Features

We check the lexical features in the sentence first. We find that the words in between play a major role in deciding the relation between the two entities. But the word varies in different cases. Sometimes it is the verb that decides the relation, sometimes it is the preposition; and other times it may be the determiners.

So, we take one word before entity one “e1”, a word after entity two “e2” and 6 words in between both the entities for our consideration and analysis of the data. We are also taking the POS tags of the above.

2. Dependency Features

   Next, we check the dependency features on the entities using Spacy. Here, we check the dependency of the nominals with the root verb and try to find a relation between the entities through the root verb. We took the head of the entities and their dependency relation with the entities as the feature

3. Synsets

   We are also finding the hypernyms, hyponyms, meronyms and holonyms (NLTK-wordnet) of the entities. Finally, we are using Spacy to get the named entities of the entities.

4. Name-Entity Relation

Finally, we are taking the name entity relation of the entities using Spacy

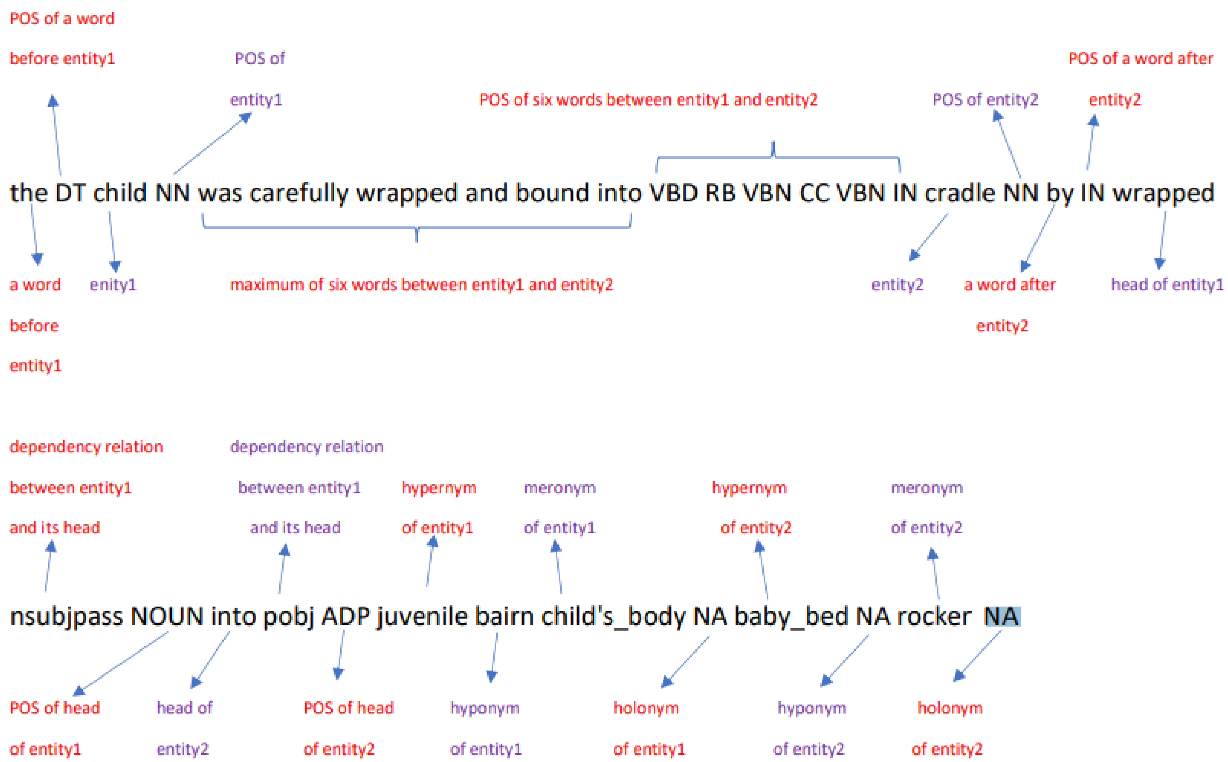
*For features will null value or features where we are unable to get a value we are using “NA” as the default vale for them.

Below is an example of the feature extracted for a sentence

Input Sentence: "The <e1>child</e1> was carefully wrapped and bound into the <e2>cradle</e2> by means of a cord."

Feature Vector:

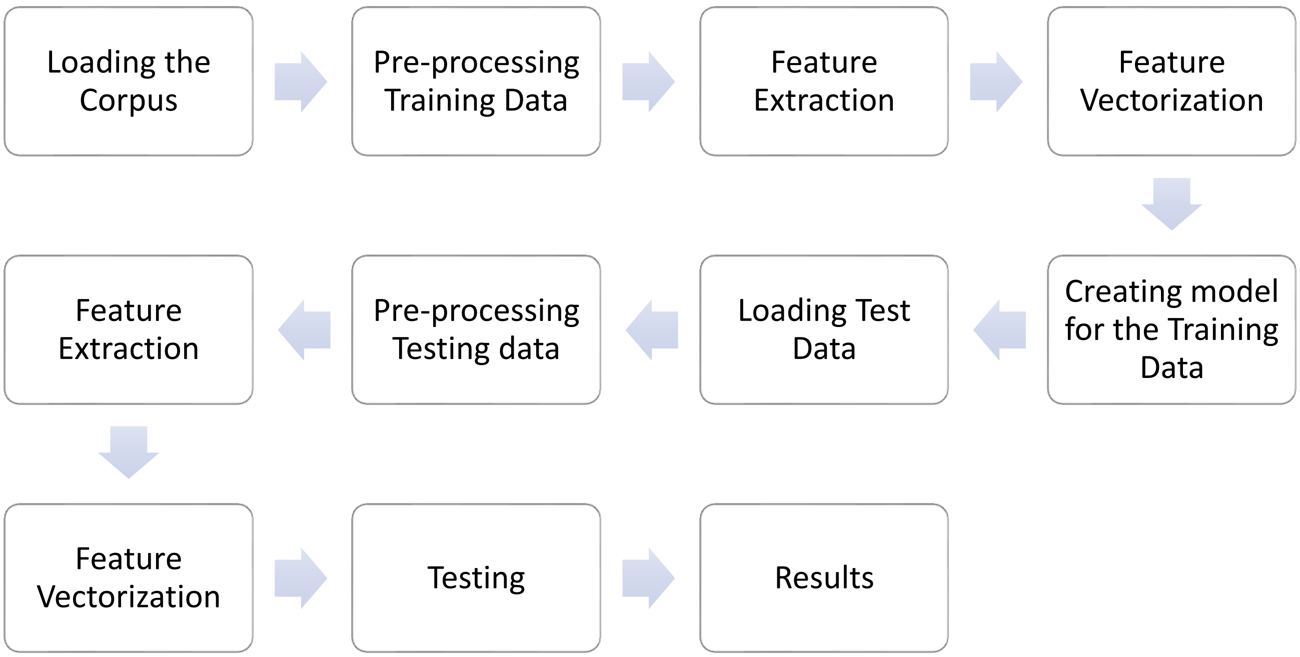
the DT child NN was carefully wrapped and bound into VBD RB VBN CC VBN IN cradle NN by IN wrapped nsubjpass NOUN into pobj ADP juvenile bairn child's_body NA baby_bed NA rocker NA NA



Step2: Creating Machine Learning Model

We are using Naïve Bayes model from SKLearn Library. Since the extracted feature is a textual data, we are converting it to numeric value using the TF-IDF vectorization. Finally, we are feeding the vectorized feature vector to our Naïve Bayes Machine Learning Model.

Algorithm Architecture:



Calculations:

For a confusion matrix with actual values vs predicted values, the precision and recall is calculated by:

Precision:
Fraction of docs assigned class *i* that are actually about class *i*: $$\frac{c_{ii}}{\sum_j c_{ji}}$$

Recall:
Fraction of docs in class *i* classified correctly: $$\frac{c_{ii}}{\sum_j c_{ij}}$$

And F-score is given by: $\frac{2PR}{(P+R)}$

Example below shows an illustration of the calculations:

	Predicted			
Actual	cat	dog	pig	Count
cat	2	2		4
dog	1	1	1	3
pig		2	1	3

	cat	dog	pig	Macro Averaging
Precision	0.6667	0.2000	0.5000	0.455556
Recall	0.5000	0.3333	0.3333	0.388889
F-Score	0.5714	0.2500	0.4000	0.407143
Accuracy	40.00%			

D. Results and Analysis

a. Details of Training process:

```
<--------------------------- Training Data Details  :---------------------------------------->

Total time taken for training :  4809  seconds.
Classes                       Count
-----------------------------------------
Other                            1299
Entity-Destination(e1,e2)        763
Cause-Effect(e2,e1)              606
Instrument-Agency(e2,e1)         372
Content-Container(e1,e2)         334
Component-Whole(e1,e2)           424
Entity-Origin(e1,e2)             518
Message-Topic(e1,e2)             444
Entity-Origin(e2,e1)             138
Product-Producer(e2,e1)          359
Cause-Effect(e1,e2)              319
Member-Collection(e2,e1)         549
Message-Topic(e2,e1)             134
Product-Producer(e1,e2)          288
Component-Whole(e2,e1)           438
Content-Container(e2,e1)         153
Member-Collection(e1,e2)         70
Instrument-Agency(e1,e2)         91
Entity-Destination(e2,e1)        1

Classes                                      Precision                    Recall                    F-Score
--------------------------------------------------------------------------------------------------------------
Other                            0.21526157947350882        0.9946112394149346         0.3539241199835639
Entity-Destination(e1,e2)        0.961335676625659          0.7169069462647444         0.8213213213213213
Cause-Effect(e2,e1)              0.8436830835117773         0.6501650165016502         0.7343895619757688
Instrument-Agency(e2,e1)         1.0                        0.013440860215053764        0.026525198938992044
Content-Container(e1,e2)         0.9230769230769231         0.1437125748502994         0.24870466321243523
Component-Whole(e1,e2)           0.9047619047619048         0.04481132075471698        0.0853932584269663
Entity-Origin(e1,e2)             0.9876543209876543         0.15444015444015444        0.26711185308848084
Message-Topic(e1,e2)             1.0                        0.02252252252252252        0.04405286343612334
Entity-Origin(e2,e1)             0.0                        0.0                        0.0
Product-Producer(e2,e1)          0.0                        0.0                        0.0
Cause-Effect(e1,e2)              1.0                        0.07210031347962383        0.13450292397660818
Member-Collection(e2,e1)         0.967741935483871          0.1092896174863388         0.19639934533551553
Message-Topic(e2,e1)             0.0                        0.0                        0.0
Product-Producer(e1,e2)          0.0                        0.0                        0.0
Component-Whole(e2,e1)           1.0                        0.0182648401826484         0.03587443946188341
Content-Container(e2,e1)         0.0                        0.0                        0.0
Member-Collection(e1,e2)         0.0                        0.0                        0.0
Instrument-Agency(e1,e2)         0.0                        0.0                        0.0
Entity-Destination(e2,e1)        0.0                        0.0                        0.0
-----------------------------------------------------------------------------------------------------------
Overall                          0.5159744959958578         0.15475081084803616        0.1551683973240873
Accuracy Score ->  34.054794520547944
```

We found that our algorithm took around 80 minutes to complete the process. Our training accuracy is 34.054 %. We calculated the precision which was 51.59% and recall which was 15.47%. Our F-score was observed as 0.155168. For the calculation of precision, recall and F-score we used macro averaging method.

b. Details of Testing process:

```
<--------------------------- Test Data Details  :----------------------------------------->

Total time taken for testing :  534  seconds.
Classes                       Count
------------------------------------------
Other                            142
Entity-Destination(e1,e2)        81
Cause-Effect(e2,e1)              65
Instrument-Agency(e2,e1)         43
Content-Container(e1,e2)         41
Component-Whole(e1,e2)           50
Entity-Origin(e1,e2)             58
Message-Topic(e1,e2)             46
Entity-Origin(e2,e1)             11
Product-Producer(e2,e1)          41
Cause-Effect(e1,e2)              30
Member-Collection(e2,e1)         64
Message-Topic(e2,e1)             11
Product-Producer(e1,e2)          38
Component-Whole(e2,e1)           42
Content-Container(e2,e1)         18
Member-Collection(e1,e2)         8
Instrument-Agency(e1,e2)         11
Entity-Destination(e2,e1)        0

Classes                                     Precision                  Recall                    F-Score
--------------------------------------------------------------------------------------------------------------
Other                            0.2002840909090909        0.9929577464788732        0.33333333333333337
Entity-Destination(e1,e2)        0.8775510204081632        0.5308641975308642        0.6615384615384615
Cause-Effect(e2,e1)              0.8947368421052632        0.5230769230769231        0.6601941747572816
Instrument-Agency(e2,e1)         0.0                       0.0                       0.0
Content-Container(e1,e2)         1.0                       0.0975609756097561        0.17777777777777776
Component-Whole(e1,e2)           0.0                       0.0                       0.0
Entity-Origin(e1,e2)             1.0                       0.034482758620689655       0.06666666666666667
Message-Topic(e1,e2)             0.0                       0.0                       0.0
Entity-Origin(e2,e1)             0.0                       0.0                       0.0
Product-Producer(e2,e1)          0.0                       0.0                       0.0
Cause-Effect(e1,e2)              0.0                       0.0                       0.0
Member-Collection(e2,e1)         1.0                       0.046875                  0.08955223880597014
Message-Topic(e2,e1)             0.0                       0.0                       0.0
Product-Producer(e1,e2)          0.0                       0.0                       0.0
Component-Whole(e2,e1)           0.0                       0.0                       0.0
Content-Container(e2,e1)         0.0                       0.0                       0.0
Member-Collection(e1,e2)         0.0                       0.0                       0.0
Instrument-Agency(e1,e2)         0.0                       0.0                       0.0
Entity-Destination(e2,e1)        0.0                       0.0                       0.0
-----------------------------------------------------------------------------------------------------------
Overall                          0.27625399741236206       0.12365653340650591       0.11050348071552728
Accuracy Score ->  28.375
```

Our testing accuracy is 28.375%. We calculated the precision which was 27.62% and recall which was 12.36%. Our F-score was observed as 0.11050. For the calculation of precision, recall and F-score we used macro averaging method.

E. Summary

We calculated the counts of each classes and checked the accuracy of our training and testing data individually.

Our accuracy is as below:

- Training Accuracy: 34.054
- Testing Accuracy: 28.375

Precision, Recall and F-Score of training and testing data

	Precision	Recall	F-score
Training	0.515974	0.154751	0.155168
Testing	0.276254	0.123657	0.110503

Problems Encountered:

One of the main problems that we came across was the huge time it took to train on our dataset. Apart from this there were few issues with the training and testing dataset provided which lead to too much pre-processing time. So, we created a new dataset from the original dataset and cleaned the dataset to avoid some repetitive unnecessary processing to save time and use the new dataset to train our model.

Pending Issues:

The wordnet sometime did not return anything for hypernyms, holonyms, meronyms and hyponyms. Similarly, Spacy returned no value for NER for some sentences. For unfound or null such cases we are giving the value as "NA" which might cause unnecessary noise in the model. This issue needs to be solved

Potential Improvements:

We did not find the accuracy of the model to be satisfactory. We felt we could improve the model more by finder better ways to relate the entities and adding more features in the feature vector.

F. References

1. Bryan Rink and Sanda Harabagiu, Classifying Semantic Relations by Combining Lexical and Semantic Resources, Human Language Technology Research Institute University of Texas at Dallas Richardson, Texas
2. Dan Jurafsky || Stanford University, Text classification evaluation, https://www.youtube.com/watch?v=Wq0taCUCSlA&list=PLLssT5z_DsK8HbD2sPcUIDfQ7zmBarMYv&index=30&ab_channel=ArtificialIntelligence-AllinOne
3. Scikit-learn module on Precision, Recall and F-score calculation; https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html