Relation Extraction using a Linear SVM

Kavin Kuppusamy
University of Texas at Dallas
Richardson, TX
kxk190026@UTDallas.edu

Derek Carpenter
University of Texas at Dallas
Richardson, TX
dxc190017@UTDallas.edu

Abstract. This report presents a computational linguistics approach for extraction of entity relations. First, data preprocessing and cleaning is performed on our dataset. After preprocessing, relevant features are extracted for each of the entities. These features include word tokens, word prefixes, shortest paths between root and entities, hypernyms, part of speech tags, and more relevant information that helps the machine better understand the sentence. A feature vector extraction method is used to create our input to our machine learning model. We apply a Linear Support Vector Machine (SVM) to build our model. Finally, we test this model using unseen entities and evaluate our model based on metrics such as f-score, precision, recall, and accuracy. We found our model predicted relations with an accuracy of 75%.

1 Introduction

The purpose of this project is to design a system that can accurately extract relations between named entities. This is relevant in modern day Natural Language Processing and can be useful for extracting higher level information and context from a corpus. These relations can be things such as cause-effect, instrument-agency, and other relations that show how two entities are related. Our goal is to develop a model that can learn how these relations are specified, based on extracted features relevant to the sentence. We will use a dataset containing over 10,000 tagged sentences, each specified with 2 unique entities and their corresponding relation. Using this dataset, we can train our SVM to understand how these relations are formed and use the model to predict unknown relations between entities.

This report will go over our process of exploring how to perform these feature extraction and model development tasks. We will evaluate the relevancy of features that are extracted from sentences by how it may improve our model. We will explore the use of various supervised machine learning models to determine which one performs best at extracting relations. We will evaluate the model performances using statistical measures such f-score, precision, recall, and accuracy. Finally, we will show how we tuned our model's hyperparameters to increase performance and select a final product.

After exploring various methods of designing this system we came to a design which can be seen in the architectural diagram in Figure 1.

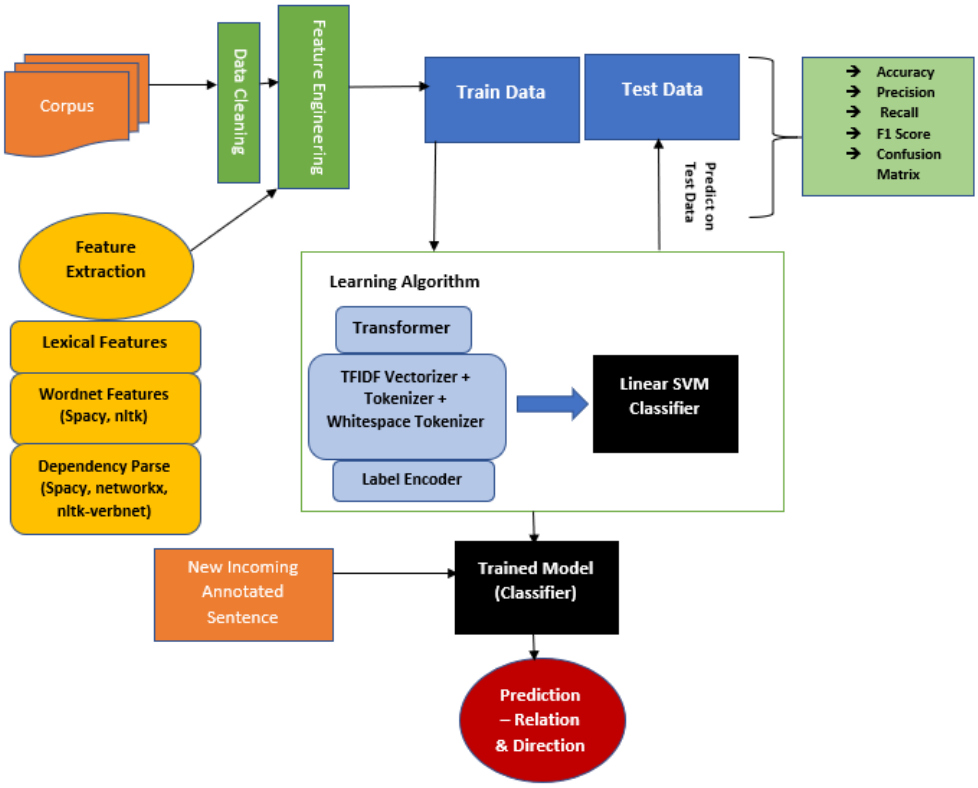

Figure 1: Architectural Diagram

Here we see the process flow of our system. Starting with an initial corpus of textual data, we preprocess and clean this data into something our system can efficiently utilize. We then perform feature engineering to extract relevant features. Using these extracted features, we perform a transformation of our features into something that can be understood by our model, using tools such as the TFIDF Vectorizer. The Linear SVM is then trained using this data and will be tested against the unseen test data. The model will be evaluated on relevant metrics and tuned to maximize performance. Finally, with the system's design completed, we can use this model to predict relations on any input test sentence.

2 Data Processing

Our dataset must be preprocessed and structured in such a way that our program can utilize it efficiently. Here we will discuss how we extract and structure the entities (sentences) from the dataset. We will also discuss the distribution of the entities and their corresponding classes.

2.1 Dataset

We have been provided a corpus with 10,717 entities. Each entity contains a sentence as seen in Figure 2.

```
14      "The <e1>burst</e1> has been caused by water hammer <e2>pressure</e2>."
Cause-Effect(e2,e1)
```

Figure 2: Example of Dataset Entity

Each sentence contains exactly two named entities as defined by the tags <e1> ENTITY1 </e1> and <e2>ENTITY2</e2>. Each of these sentences is labeled with a relation type. As seen in Figure 2, this sentence has a relation of Cause-Effect for Entity2 to Entity1.

The dataset contains 19 classifications for our sentences which can be seen in Table 1.

Table 1: Relation Classifications and distribution of classes

Class	# Entities	% of Dataset
Component-Whole e2, e1	621	5.79%
Other	1864	17.39%
Instrument-Agency e2, e1	541	5.05%
Member-Collection e1, e2	110	1.03%
Cause-Effect e2, e1	853	7.96%
Entity-Destination e1, e2	1135	10.59%
Content-Container e1, e2	527	4.92%
Message-Topic e1, e2	700	6.53%
Product-Producer e2, e1	517	4.82%
Member-Collection e2, e1	813	7.59%
Entity-Origin e1, e2	779	7.27%
Cause-Effect e1, e2	478	4.46%
Component-Whole e1, e2	632	5.90%
Message-Topic e2, e1	195	1.82%
Product-Producer e1, e2	431	4.02%
Entity-Origin e2, e1	195	1.82%
Content-Container e2, e1	205	1.91%
Instrument-Agency e1, e2	119	1.11%
Entity-Destination e2, e1	2	0.02%

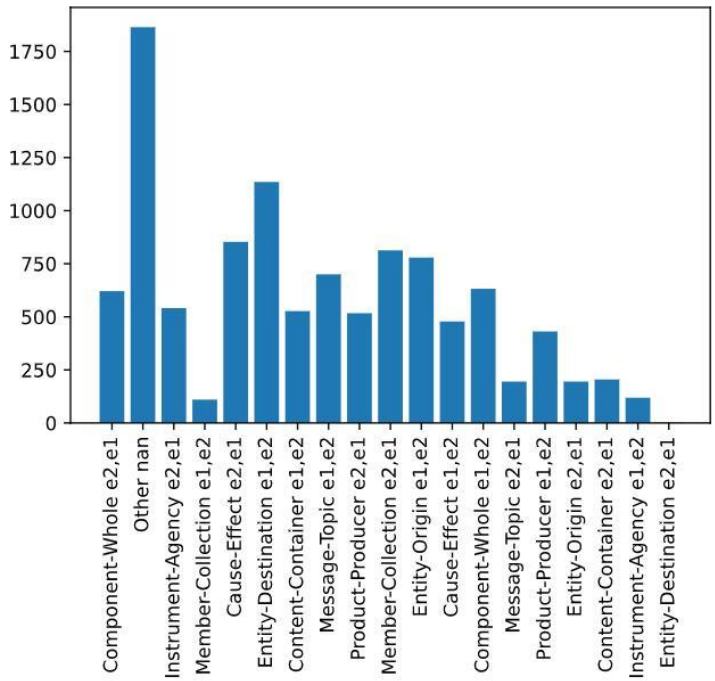

Figure 3: Distribution of dataset classes.

The dataset is not following an equal distribution among classes. The 'Other' class is heavily favored as a percentage of this dataset accounting for 17.39%. This is because the 'Other' class refers to two entities that are either not related or are some other classification that is not included in our 18 other classes. The remaining entities account for 1-10% of the dataset, which is more inline with a proper dataset needed for classification.

2.2 Data Processing

Now that we have a clear understanding of the dataset, we can begin our data processing. For each of these sentences we will process the sentence into an object. We will use python and various Natural Language Processing open sources libraries to extract this information.

First, we will read each line from the dataset and use the Regular Expressions seen below to filter the sentences. Here we will extract each named entity and the rest of the words in the sentence.

$$E1 = (?<=<e1>).*(?=</e1>)$$
$$E2 = (?<=<e2>).*(?=</e2>)$$
$$before\_E1 = (?<=").*(?=<e1>)$$
$$between\_E1\_E2 = (?<=</e1>).*(?=<e2>)$$
$$after\_E2 = (?<=</e2>)[^"]*$$

After each of these Regular Expressions have been passed through our line, we can concatenate them to form the full, cleaned sentence.

$$Sentence = before\_E1 + E1 + between\_E1\_E2 + after\_E2$$

We will create an object in python called "sentence" which will be used to store all of our features related to each sentence entity. For now, we have just extracted the raw data from our dataset, the sentence itself, and the named entities.

With our sentences processed, we will designate a portion of the dataset to training and test sets that will be used to train and evaluate our model. Table 2 shows the distribution of test and train data from our dataset.

Table 2: Test/Train Split

	Entries	**Percentage**
Test	2680	0.25
Train	8037	0.75

This is a common standard in machine learning applications to split the data in such a distribution. Using 75% of the entities to train our model and 25% to test the model. The test data will never be seen by the model during the training phase. This allows us to properly test our model to ensure that it will perform well when it sees a sentence that is not classified.

With the dataset processed, we can begin our feature extraction from the data.

2.3 Features

We trained our model with 21 features that belong to categories such as lexical features, wordnet features like hypernyms, and information about the sentence's dependency parse tree. The following explains each feature.

Lexical Features

1. **Entities:** We extracted the annotated nominals from the given sentences. If the entity is composed of more than one word, we parse the noun phrase and take the head word. (eg., San Jose University -> University).
2. **Entity Tag:** Using NLTK POS tagger, we attached Part of Speech Tag for each entity.
3. **Words between nominals:** Using the words between the nominals is a highly relevant feature regarding a relation in a sentence.
4. **Prefix of Length 5:** Using the prefixes of length 5 for the words between the nominals provides a kind of stemming (produced -> produ, caused -> cause). This stemming effect

reduces the complexity of our features and generalizes the words, producing a better generalization of other potential similar sentences.

5. **Distance between nominals:** The number of words between the two entities mentioned. This feature will mainly help the model predict the classes such as Product-Producer and Entity-Origin because these relations usually would not have intervening tokens. For example, organ builder, Chestnut flour.
6. **POS Tag Between:** We extract a coarse-grained part of the speech sequence for the words between the nominals. This feature is produced by extracting the first letter of each token's POS tag. This feature is included mainly to find the class such as Member-Collection which usually invoke prepositional phrases such as : of, in the and its corresponding POS TAG Between I and 'I_D' respectively.
7. **Words before E1, After E2, Before E1, and Single word after E2:** Extracting all the tokens before the e1 entity, after the e2 entity, and all the tokens before e1 and a single token after e2.

WordNet Features

1. **E1 and E2 Hypernyms:** Using the wordnet package, we extract hypernyms for both e1 and e2.
2. **Lowest Common Hypernym:** We utilize Wordnet's lowest_common_hypernyms method to extract the lowest common hypernym between the two named entities. It will perform a search of hypernyms until there is a potential match between the two.
3. **Wu-Palmer Similarity:** It calculates relatedness by considering the depths of the two synsets in the WordNet taxonomies, along with the depth of the LCS (Least Common Subsumer). The score can be 0 < score <= 1. This feature mainly helps in finding how similar are the given two nominals.

Dependency Parsing

Motivated from the paper (A Shortest Path Dependency Kernel for Relation Extraction), If e1 and e2 are two entities mentioned in the same sentence such that they are observed to be in a relationship R, the contribution of the sentence dependency graph to establishing the relationship R(e1, e2) is almost exclusively concentrated in the shortest path between e1 and e2 in the undirected version of the dependency graph.

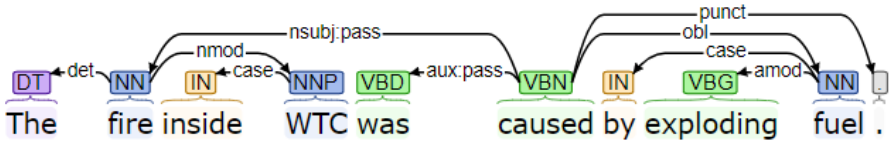

Figure 4: Dependency Parse for 'The fire inside WTC was caused by exploding fuel.'

1) **Dependency Path length1**: Using Spacy and networkx, produced the dependency parse tree and find the shortest path dependency between two entities. This feature mainly helps in finding the parent node which connects the two entities.
2) **Dependency Path length2:** Encode the complete path between e1 and e2 including dependency features.

3) **Connecting Path:** Finding the tokens between the two entities from the path extracted by shortest path dependency.
4) **SDP Root Node Lemma:** This feature will extract the lemmatized head word of the phrase/token connecting two entities in the two entities.
5) **Shortest Path Length:** This feature will reveal that the shortest path length given the dependency parse.

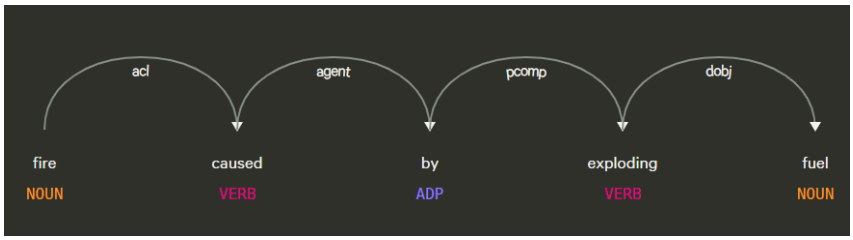

Figure 5: Shortest Path between fire and fuel and length is 4

6) **Root Word Location:** This feature will give the location (BEFORE, BETWEEN, AFTER) of the root word between two entities, found from dependency parse

Table 3 shows the features extracted from the sentence "The fire inside WTC was caused by exploding fuel.".

Table 3: Example Set of Extracted Features

Features		Examples
Lexical Features	e1	fire
	e2	fuel
	before_e1	The
	after_e2	.
	word_outside	[The, .]
	prefix5between	[insid, WTC, was, cause, by, explo]
	distance_between	6
	e1_tag	NN
	e2_tag	NN
Wordnet Features	HyperE1	[bake.v.01,passion.n.01,chase_away.v.01,combustion.n.01,element.n.05,happening.n.01,trial.n.06,criticism.n.01,remove.v.02,destroy.v.02,supply.v.01,blast.v.07,fireplace.n.01,attack.n.01,fuel.n.01,make.v.03]
	HyperE2	[stimulate.v.03, take_in.v.12, substance.n.07,supply.v.01]
	lc_hyper	entity.n.01
	max_entity_sim	1
Dependency Parse	dep_path_len1	[cause->nsubjpass->E1, cause->dobj->E2]
	dep_path_len2_location	E1_nsubjpass_BETWEEN_dobj_E2
	connecting_path	caused by exploding
	sdp_root_lemma	cause
	e1_dep	nsubjpass
	e2_dep	dobj
	shortest_path	[fire, caused, by, exploding, fuel]
	shortest_path_len	4
	root_word_location	BETWEEN

3 Model Experiment and Analysis

Model Experiment

After we preprocessed the data and extracted all the features. We used Sklearn TF-IDFVectorizer (Term Frequency Inverse Document Frequency). This algorithm will transform the text into a meaningful representation of numbers which is used to fit the machine algorithm for prediction. We used Sklearn Pipeline to apply Vectorizer and Transformer on different features. The TFIDF Vectorizer along with whitespace Tokenizer and other types of tokenizers were used to Vectorize and Transform the feature values. We used LabelEncoder for predictor, which is the concatenation of direction and relation. With our feature vector structured in a manner that machine learning algorithm we are ready to implement a model that can learn from our features.

We explored various machine learning algorithms such as Neural Networks, Random Forest Classifiers, Decision Trees, and Support Vector Machines. For our implementation we found that a Linear Support Vector Machine (SVM) performed the best at classifying our relations. A support vector machine is a type of machine learning technique that learns a linear separator for data based on some support vector. It is used primarily for classification purposes and is sometimes preferred over iterative models such as a neural network which learns model weights through a series of epochs. We utilized the Scikit-Learn linear SVM and implemented it with liblinear, which helps with adding weight penalties as well as working well with a dataset that has a large number of entities.

Experiment Results and Analysis

After training our model, we performed a test to see how it performs using the unseen, classified, test data. Table 4 shows the precision, recall, and f-score, for each class in our dataset.

Table 4: Performance of SVM on Test Dataset

Relation	precision	recall	f1-score
Cause-Effect e1,e2	0.95	0.88	0.91
Cause-Effect e2,e1	0.85	0.88	0.86
Component-Whole e1,e2	0.80	0.81	0.80
Component-Whole e2,e1	0.71	0.74	0.73
Content-Container e1,e2	0.79	0.92	0.85
Content-Container e2,e1	0.86	0.82	0.84
Entity-Destination e1,e2	0.83	0.89	0.86
Entity-Origin e1,e2	0.81	0.88	0.84
Entity-Origin e2,e1	0.75	0.92	0.83
Instrument-Agency e1,e2	0.72	0.60	0.65
Instrument-Agency e2,e1	0.70	0.77	0.73
Member-Collection e1,e2	0.82	0.50	0.62
Member-Collection e2,e1	0.77	0.94	0.84
Message-Topic e1,e2	0.77	0.87	0.82
Message-Topic e2,e1	0.78	0.63	0.70
Other	0.53	0.38	0.44
Product-Producer e1,e2	0.73	0.79	0.76
Product-Producer e2,e1	0.70	0.67	0.69
macro avg	0.77	0.77	0.77
weighted avg	0.74	0.76	0.74

Table 5: Accuracy with Feature Addition

	Accuracy
Lexical features	65%
Wordnet - hypernyms	72%
Wordnet - LC hypernyms	73%
Dependency Parse	75.5%

Table 6: Time Results

	Minutes
Feature Extraction	18:00
Model Training	0:30
Relation Prediction	0:06

Because our approach makes use of many different features, we ran ablation tests on each set of features mentioned in Table 1 to determine which types of features contributed the most to classifying semantic relations. We further experimented with verbnet features like root word Levin class which did not help the model accuracy. After testing the model on test data, we got an accuracy of 75.5% and macro average precision of 77%. To further understand the performance of our model we generated a confusion matrix, shown in Figure 6, which provides a good visualization of how the model is performing on each class.

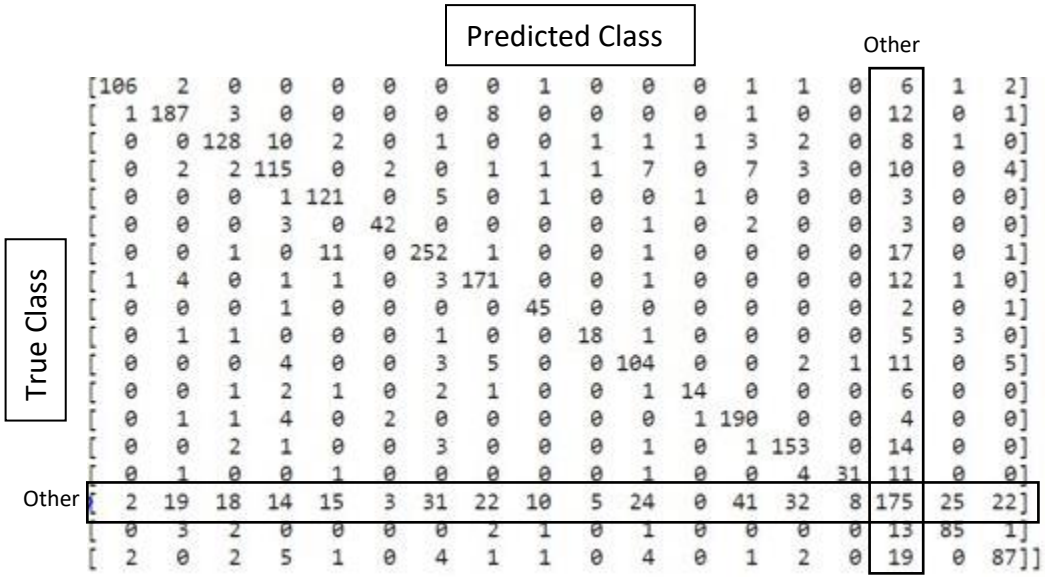

Figure 6: Confusion Matrix of Testing Model on Test Dataset

With the Confusion Matrix attained, we see that precision on 'Other' class is low because of the dataset having more bias towards the '*Other'* class (17.3%). There were significantly more incorrect predictions relating to this class. Our model attempts to classify a 'Other' tagged relation to one of the known classes, which can be seen from the 'Other' *row* in the confusion matrix. Similarly, our model incorrectly predicts Other in the frequency shown in the 'Other' *column* in the matrix. We do not believe this is a major issue as it correctly classifies the relations that are not labeled 'Other' with a high frequency. With our model performing tell on test data, we can utilize this model for prediction of relations sentences for future applications, such as semantic knowledge extraction.

4 Conclusion and Future Work

In conclusion, we have utilized a Linear Support Vector Machine to build a model that correctly predicts relations between named entities in various sentences. We combined diverse lexical, syntactic, semantic, and structural features to best represent our sentences. Due to the imbalanced dataset and bias towards one single class, the model accuracy was weighed down by the performance of the 'Other' class. We will apply future experiments with the model by balancing the dataset, applying a Random Under sample on Majority class, and by applying balancing weights on the classifier.

5 References

1) Rink, Bryan, and Sanda Harabagiu. "Utd: Classifying semantic relations by combining lexical and semantic resources." In Proceedings of the 5th International Workshop on Semantic Evaluation, pp. 256-259. 2010.
2) Szarvas, György, and Iryna Gurevych. "TUD: semantic relatedness for relation classification." In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pp. 210-213. 2010.
3) Negri, Matteo, and Milen Kouylekov. "Fbk_nk: A wordnet-based system for multi-way classification of semantic relations." In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pp. 202-205. 2010.
4) Esuli, Andrea, Diego Marcheggiani, and Fabrizio Sebastiani. "ISTI@ SemEval-2 Task 8: Boosting-Based Multiway Relation Classification." In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pp. 218-221. 2010.
5) Beamer, S. Bhat, B. Chee, A. Fister, A. Rozovskaya, and R. Girju. 2007. UIUC: a knowledge-rich approach to identifying semantic relations between nominals. In ACL SemEval07 Workshop
6) Hendrickx, S.N. Kim, Z. Kozareva, P. Nakov, D. ´O S´eaghdha, S. Pad´o, M. Pennacchiotti, L. Romano, and S. Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals.