Problem Description

The goal of this project was to classify the semantic relation between a pair of entities in a sentence, specifically the Cause-Effect, Instrument- Agency, Product-Producer, Content-Container, Entity-Origin, Entity-Destination, Component- Whole, Member-Collection, Message-Topic relations, as well as identify the direction of the relation between the entities, i.e. whether the relation is from entity1 (e1) to entity2 (e2) or vice-versa or whether the relation is *other*, which is a special case that does not have any direction. Different syntactic, lexical and contextual features had to be extracted in order to train models for the classification tasks.

Proposed Solution

The identification of the entity relations and the identification of the relation directions were separated into two different classification tasks. A Voting Classifier model that combines the predictions of a decision tree, a random forest and a Guassian Naive Bayes model was trained to classify the semantic relation present between the two entities in a sentence. Then a separate Voting Classifier model that also combines the predictions of a decision tree, a random forest and a Guassian Naive Bayes model was trained to specify the direction of the relation.

The SemEval dataset was used as the training corpus for the models. The training dataset consists of 8000 annotated sentences and the test set consists of 2717 examples. First, a class *CorpusReader* was created to read the training data from the training corpus and extract the sentences, the relations, the directions of the relations and the entities themselves.

The extracted data was then used for obtaining the following features of each sentence:

- **Tokens** or words in the sentence
- **Lemma** of each word in the sentence
- **Part of Speech (POS) Tag** of each word in the sentence
- **Dependency Parse Tag** of each word in the sentence and its head
- **Dependency Parse Part of Speech Tag** of the head of each word in the sentence
- **Shortest Dependency Path** between entities if direction is from e1 to e2, else e2 to e1
- **Length of the Shortest Dependency Path** between e1 and e2
- **Hypernym, Hyponym**, **Meronym**, **Holonym** of each entity
- **WordNet Path Similarity** between the synsets of e1 and e2
- **Named Entity Recognition (NER) Tag** of each entity
- **Stem** of each entity
- **Prefixes of length 5** of the words between e1 and e2
- **Word before e1**
- **Word after e2**
- **POS Tags** of the words between e1 and 2
- **Count of words** between e1 and e2
- **Gloss:** Given a pair of annotated nominals, e1 and e2, these features are set to 1 every time either e1 appears in the gloss of e2, or vice-versa.

The features were then numerically encoded. Different combinations of the encoded features were integrated into different two-dimensional feature vectors; in each vector, each column consisted of a feature and each row consisted of the corresponding features of each sentence.

The classification model we adopted in our project splits the two tasks of classifying relation and direction of relation into two different classification tasks.

A vectorial representation of features is created for each example. The different feature sets were each used to train the following set of Machine Learning models. A 10-way classifier is trained to classify the type of semantic relation present between two entities in a given sentence.

- A baseline Bag-of-Words model
- A Decision Tree model
- A Random Forest model
- A Guassian Naive Bayes model
- A Voting Classifier model that combines the predictions from the Decision Tree, Random Forest and Gaussian Naive Bayes models

After training the models on how to classify the semantic relation between two entities, the different feature sets were then used to train the following set of Machine Learning models to identify the direction of the relation:

- A Random Forest model
- A Voting Classifier model that combines the predictions from a Decision Tree, a Random Forest and a Gaussian Naive Bayes model

It was observed that for the classification of the semantic relation between two entities, the Voting Classifier model performed the best, achieving an accuracy of 61.87%. For the classification of the direction of the semantic relation, the Voting Classifier model performed the best as well, achieving an accuracy of 92.36%. Both of these classifiers were trained on the following set of features to achieve the best performance: shortest dependency path, length of the shortest dependency path, e1, e2, POS tag of e1, POS tag of e2, prefixes of length 5 between e1 and e2, word before e1, word after e2, count of the words between e1 and e2, and POS tags of the words between e1 and e2.

Implementation Details

1. Programming Tools

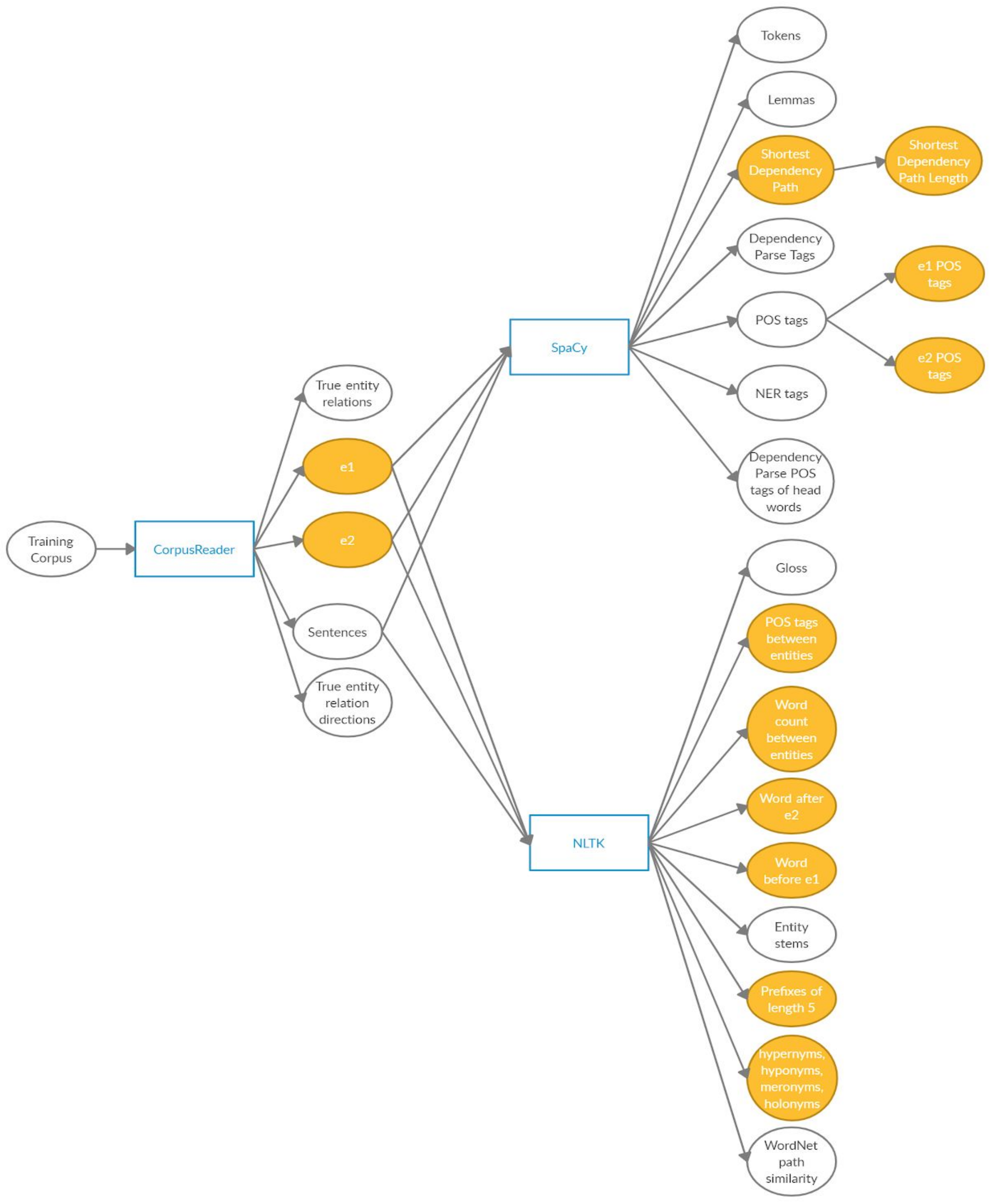
   Programming Language used:
   - Python 3

   Programming libraries used:
   - Pandas
   - spaCy

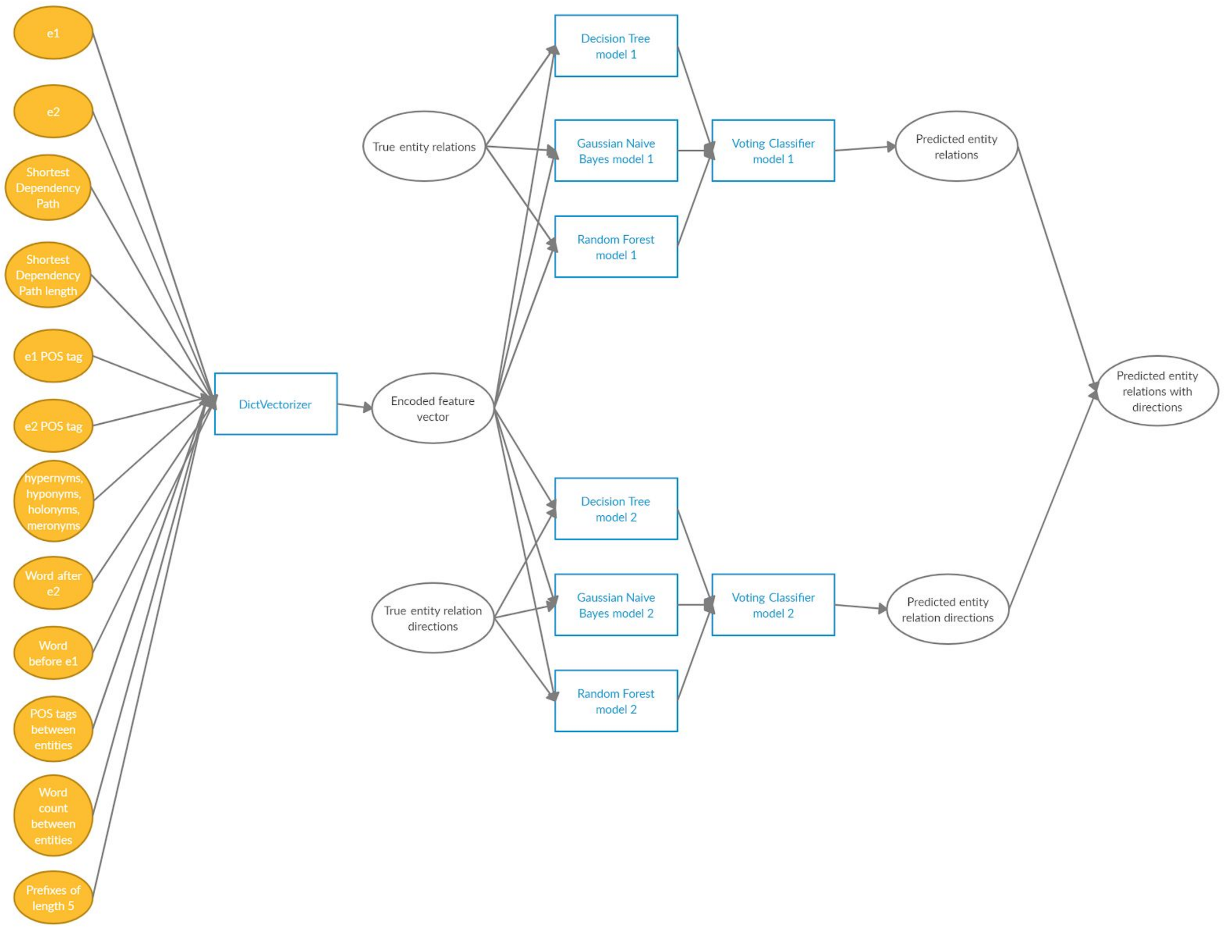
- NLTK
- Scikit-learn
- NetworkX

2. <u>Architectural Diagram</u>

Feature Extraction:



Classification of entity relations and their directions:



e1
e2
Shortest Dependency Path
Shortest Dependency Path length
e1 POS tag
e2 POS tag
hypernyms, hyponyms, holonyms, meronyms
Word after e2
Word before e1
POS tags between entities
Word count between entities
Prefixes of length 5
DictVectorizer
Encoded feature vector
True entity relations
True entity relation directions
Decision Tree model 1
Gaussian Naive Bayes model 1
Random Forest model 1
Voting Classifier model 1
Predicted entity relations
Decision Tree model 2
Gaussian Naive Bayes model 2
Random Forest model 2
Voting Classifier model 2
Predicted entity relation directions
Predicted entity relations with directions

3. <u>Results and Error Analysis</u>

   The classification model we adopted for relation achieved the best overall macro-averaged F1 score of 0.62 for just relation and 0.57 when both relation and direction predictions are combined on the test set. Time taken by the model to make predictions on the test set is 4.8 seconds for relation classifier and 1.2 seconds for direction classifier.

   The baseline Bag-of-Words model has achieved an accuracy of 29.88% with 2000 features.

   For relation classification we adopted a voting classifier that takes the majority vote of Decision Tree, Gaussian Naive Bayes and random Forest, and it achieved an accuracy of 61.87%. The results in table 2.1 show the performance of the system on a test set for relation classifier.

Relation	**Accuracy**
Cause-Effect	61.24
Component-Whole	56.00
Content-Container	62.63
Entity-Destination	61.93
Entity-Origin	63.22
Instrument-Agency	61.06
Member-Collection	90.75
Message-Topic	72.47
Product-Producer	73.61
Other	45.04
Total	61.87

Table 1: Accuracy of each relation

Metrics	
Accuracy	0.62
Macro Precision	0.65
Macro Recall	0.62
Macro F-1 Score	0.62

Table 2.1: Metrics assuming that relation is classified correctly (but not the direction)

Metrics	
Accuracy	0.60
Macro Precision	0.59
Macro Recall	0.56
Macro F-1 Score	0.57

Table 2.2: Metrics assuming that both relation and direction are classified correctly

A voting classifier has been adopted for direction classification which takes the majority vote of Decision Tree, Gaussian Naive Bayes and random Forest and achieved an accuracy of 92.36%.

The results obtained from both the classifiers are combined and final performance of the system is shown in table 2.2 with macro-averaged F1 score of 0.57 is achieved.

4. Issues encountered and resolved

The following lists the issues that were encountered in the course of the project and how they were resolved:

- Some of the pairs of entities in the training corpus had *no relation* in the pair. A *no relation* means that there was no direction of relation between the two entities either. This was resolved by assuming that the *no relation* or the *other* relation has no direction class, so the direction classifier was trained on only 2 classes, (e1,e2) and (e2,e1). The intuition behind it was that if there exists no relation between two entities then there cannot be a direction of relation between them; there can only be directions for relations that exist. Other possible solutions were thought of but not implemented due to time constraints that included creating 2 instances, i.e. 2 vectorial representations of *other* relations - *other* with the direction (e1,e2) and *other* with (e2,e1), and the correct instance would be decided based on the following:

    - If both representations result in *other* classify it as *other*
    - If one output is *other* and another is a relation *r* then classify with *r* and direction determined by the input representation
    - If the outputs of both the representations are different relations *r1* and *r2* choose the one predicted with maximum probability.

- A significant portion of the entities had multiple synsets; each entity had multiple parts of speech in WordNet and multiple synsets for each part of speech sense. The presence of multiple synsets for an entity was an issue when extracting the hypernyms, hyponyms, holonyms and meronyms of the entity because WordNet does not allow the extraction of hypernyms, hyponyms, holonyms and meronyms of a set of multiple synsets; it only allows the use of one synset.
  Therefore, in order to find the most probable synset of an entity, its part of speech based on its context was first extracted. That part of speech tag was then used to select the appropriate part of speech for the entity in WordNet. The first synset for the appropriate WordNet part of speech was then used for evaluation. For example, the set of synsets for the entity “dog” is [Synset('dog.n.01'), Synset('frump.n.01'), Synset('dog.n.03'), Synset('cad.n.01'), Synset('frank.n.02'), Synset('pawl.n.01'), Synset('andiron.n.01'), Synset('chase.v.01')]. Based on its context, if it is found that “dog” is a noun, then only the noun synsets are considered, among which the first is selected. In this case, Synset('dog.n.01') is selected.

5. <u>Pending Issues:</u>
    - A significant portion of the entities had empty synsets.
    - A significant portion of the entities had empty NER tags. Despite using different libraries (namely SpaCy and NLTK) for NER extraction, the NER tags of a considerable number of entities remained empty.
    - Most of the *other* relations were being misclassified. In fact, the *other* relations had the highest percentage of misclassifications out of all the relations, and no extra features could be found to improve their predictions. The model without the *other* relations performed better than the model with them.

6. <u>Potential Improvements:</u>
    - Gazetteers can be used to get more specific, low-level NER tags for every entity.
    - A Machine Learning model like a Naive-Bayes or a Decision Tree model can be trained with bag-of-words, collocational and other features to identify the correct synset of each entity. Such a learning-based approach will possibly lead to better

word sense disambiguation of each entity than simply using context-based part-of-speech tags.

- The paths in the dependency parse trees can be generalized by replacing the verbs in the paths with their corresponding top-level Levin class.
- Some additional features can be used for model training:
    - Pre-existing relations between the entities; TextRunner can be used to extract such relations
    - Semantic roles of the entities in the sentences that they occur in using PropBank and FrameNet parses
    - Nominalizations of the entities based on NomLex in order to generalize the entities
- Correlation between the semantic features, i.e. the WordNet features of Instrument-Agency and Product-Producer relations, was observed. For example, sentences shown in the table below show that the entity *virtuoso*: a person highly skilled in music from Instrument-Agency is related to the entity *artists:* who creates music of Product-Producer even though they are from different relations.

Texas-born <e1>virtuoso</e1> finds harmony, sophistication in Appalachian <e2>instrument</e2>.	Instrument-Agency(e2,e1)
For us the term <e1>artists</e1>' <e2>book</e2> simply means a book made by an artist; a book made as a work of art rather than as a literary artifact.	Product-Producer(e2,e1)

A similar correlation between Content-Container and Entity-Destination relations was also observed, as shown in the table below:

Flowers are nice, but don't last very long and the <e1>fruit</e1> in a fruit <e2>basket</e2> often goes bad.	Content-Container(e1,e2)
A large <e1>marble</e1> was dropped into the <e2>bowl</e2>.	Entity-Destination(e1,e2)

Features may be extracted to distinguish these relations.