Word Embeddings

A Thesis
Submitted in partial fulfillment of
the requirements for the degree of
Master of Technology
by

Deepak Singh Baghel
(Roll No. 203050005)

Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Mumbai 400076 (India)

22 June 2022

Acceptance Certificate

Department of Computer Science and Engineering

Indian Institute of Technology, Bombay

The thesis entitled "Word Embeddings" submitted by Deepak Singh Baghel (Roll No. 203050005) may be accepted for being evaluated.

Date: 22 June 2022

Approval Sheet

This thesis entitled "Word Embeddings" by Deepak Singh Baghel is approved for the degree of Master of Technology.

Examiners

Supervisor (s)

Chairman

Date:

Place:

Abstract

Ontologies play an important role in maintaining the concept hierarchies of any domain. However, ontologies require significant effort to create and maintain because of manual and time-consuming tasks. Word embeddings is popularly used in many natural language processing tasks because of its ability to cope up with semantic sensitivity. In this study, I have proposed an efficient semi-supervised ontology population method using word embeddings for FoodOn ontology. I obtained food word embeddings by training word2vec model(Mikolov *et al.*, 2013a) on "wikipedia corpus"(Youn *et al.*, 2020). I implemented a digraph data structure to minimize the number of classes visited during the FoodOn ontology population and improved time complexity by $\approx$ 88% and precision by $\approx$ 2% when compared to previous method.

keywords: FoodOn ontology, word embeddings, wikipedia corpus, ontology population.

Table of Contents

Chapter 1

Introduction

In various computational tasks use of ontologies is becoming increasingly involved. Many of the research areas such as knowledge engineering and representation, information retrieval and extraction, and knowledge management and agent systems have incorporated the use of ontologies to a greater extent. The need for efficient food systems to support food production, distribution, and nutrition to serve a growing planet is now more apparent (Youn *et al.*, 2020). I plan to explore the application of word embeddings in natural language processing problems, and in this stage, I have worked on the ontology population using word embeddings.

An ontology is a "formal and explicit specification of a shared conceptualization" (Rehurek and Sojka, 1993). Ontologies are used in many applications such as text classification, word set expansions, and information extraction, etc. Consider following food ontology Figure 1.1 for illustration, three circular nodes are classes, and the rest four nodes are seed entities of corresponding classes. We want to map the new entity "banana" to an existing ontology.

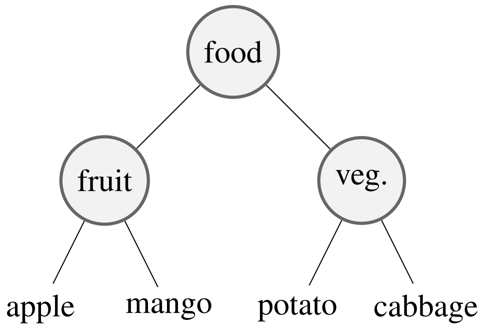

Figure 1.1: "food" is the root class of food ontology and it has two subclasses "fruit", and "veg.". "fruit" class has two seed entities apple, and mango. "veg." class also has two seed entities potato, and cabbage.

Label	Vector					
fruit	2	2	1	1	0	0
apple	1	2	1	0	0	0
mango	1	0	2	1	0	0
veg.	0	0	1	1	2	2
potato	0	0	0	1	1	2
cabbage	0	0	1	1	2	1
banana	1	2	1	0	1	0

Table 1.1: These word vectors are tuned manually for illustration. “banana” is a candidate entity that is to be mapped to either “fruit” or “veg.” based on semantic similarity.

We find the cosine similarity between candidate entity label and every class label of an ontology and map candidate entity to a class with the highest cosine similarity. Sample Label vectors are given in Table 1.1 and cosine similarity is shown in Table 1.2. “banana” is classified as fruit due to higher cosine similarity with the “fruit” class.

Class	Candidate Entity	$cos\theta$
fruit	banana	**0.41**
veg.	banana	0.17

Table 1.2: Since banana is a fruit it has a higher similarity with class “fruit” than class “veg.”. Hence banana will be classified as a fruit.

FoodOn is an ontology, a controlled vocabulary that can be used by both people and computers to name all parts of animals, plants, and fungi that can bear a food role for humans and domesticated animals, as well as derived food products and the processes used to make them.

As we have seen above to map any candidate entity to an existing ontology we are finding similarity of candidate entity with every class in ontology. This method of populating an ontology is similar to the LOVE Model that I have discussed in the next chapter briefly. Any entity can be classified into some set of classes or parent classes of the same classes. Hence considering all the classes of ontology is a time-consuming task.

1.1 Summary

This chapter has discussed the problem statement and need for solving the problem. Next chapter covers the previous work. I have discussed an efficient approach to populate FoodOn ontology in chapter 3. FoodOn ontology is a digraph. For each candidate entity, we traverse in a digraph heuristically to minimize the number of nodes visited and find scores with each class during traversal, and map the candidate entity to a class with the highest score. Our method improved time complexity by $\approx$ 88% and precision by $\approx$ 2% when compared to previous method. Chapter 4 describes application of rule-based parsing to our problem. Chapter 5 finds effect of optimal seeds entities on precision. Chapter 6 discusses about Animal Plant Ontology population. And chapter 7 concludes and discusses future work.

Chapter 2

Previous Work

There are many previous works(Youn *et al.*, 2020)(Jayawardana *et al.*, 2017)(Karadeniz and Ozgur, 2019) which are using word embedding to populate ontologies of various domains, for example, legal, medical, food, etc. Every method considers all the classes of an ontology to classify a candidate entity.

2.1 LOVE Model

The LOVE Model(Youn *et al.*, 2020) is shown in Figure 2.1. This model takes "Skeleton Ontology" and candidate entities as input and populate "Skeleton Ontology" with candidate entities. "Skeleton Ontology" is an ontology with all the classes and at most two seed entities in each class, the rest of the entities of classes are called candidate entities. There are a total of 2764 classes and 10,865 instances in FoodOn Ontology out of which 7745 instances are randomly generated as candidate entities and rest entities are seeds.

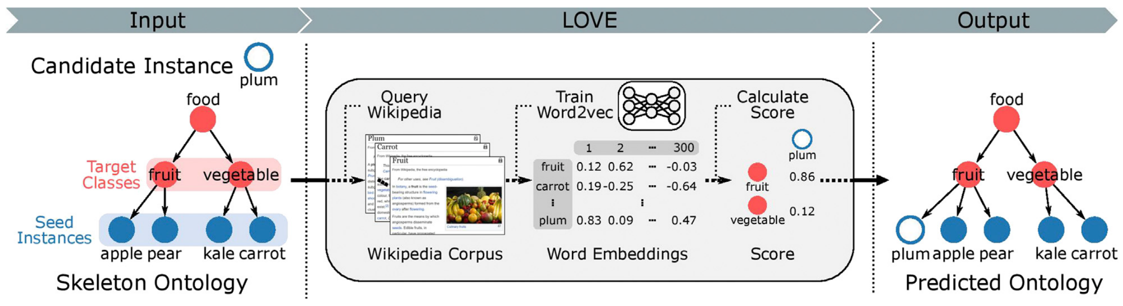

Figure 2.1: Modules of LOVE Model. There are two target classes "fruit" and "vegetable". The fruit class has two seed entities namely apple and pear for vegetable class seed entities are kale and carrot. there is one candidate entity "plum". "plum" is classified as a fruit by the LOVE Model. Figure reprinted from (Youn *et al.*, 2020).

For a class with more than two entities two entities are taken as seed entities and for a class with exactly two entities one is taken as seed and one as non-seed, and for a class with one entity same is taken as seed entity.

LOVE Model generates a digraph from FoodOn ontology and creates Skeleton Ontology. It extracts all the words from class labels and entity labels for querying wikipedia to create a wikipedia corpus. Preprocessed the corpus as follows: lower-case conversion, punctuation stripping, white-space stripping, numeric stripping, stop-word removal, short words stripping, and lemmatization.

For training, gensim (Rehurek and Sojka, 2010) implementation of the word2vec skip-gram model (Mikolov *et al.*, 2013a) is used. Default settings of the gensim word2vec model are used except for the following parameters: number of epochs of 100, window size of 5, and minimum count of 1, and vector dimension of 300.

let i be a food instance and c be a target class, then $i \in I$ and $c \in C$, where I is the group of all food instances we seek to map and C is the group of target classes to which we map the food instance. We also define I_c to be all the food instances within a class c. To map the instance to its appropriate target class, this method propose an approach based on the similarity of word embeddings. their criteria for optimal population consider a linear combination of two scores:

$$score(c,i) = \alpha \cdot score_{sibling}(c,i) + (1-\alpha) \cdot score_{parent}(c,i),$$

where α controls the ratio of the two terms. $score_{siblings}$ is the similarity of the food instance i with the seed instances in I_c. α is set to 0.8 empirically after testing all values between 0.0 and 1.0 with an interval of 0.1.

$$score_{sibling}(c,i) = sim\left(\vec{i}, \sum_{i' \in I_c} \frac{\vec{i'}}{|I_c|}\right)$$

where $\vec{\cdot}$ is the word embedding vector created by taking the average of the constituent word embeddings, $|I_c|$ is the number of all the seed instances in I_c, and sim () is the measure of similarity between the two word embedding vectors. $score_{parent}$ is the similarity of the food instance i with the target class c.

$$score_{parent}(c,i) = sim\left(\vec{i}, \vec{c}\right)$$

Finally, predicting which target class $\bar{c}$ the food instance i will get mapped to is be formulated as follows:

$$\bar{c} = \arg\max_{c} score(c, i)$$

Label vectors for each class/entity are achieved by averaging the word vectors of individual words in a label with 15% more weightage given to nouns since class/entity labels are noun phrases. Words not in the vocabulary are ignored from the class/entity label there are total 434 unique words which are ignored out of 4139 unique words in the vocabulary. Entities with zero label vectors are not considered during precision calculation also classes without entity is not considered during precision calculation. Entity has zero label vector if there is no common words in its label and vocabulary. For each candidate entity, score is calculated against every class in ontology and the candidate entity is mapped to the class with the highest score. Entities which are correctly classified are True Positive otherwise False Positive.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

LOVE model achieved precision of 34%. 3748 is the True Positive and 7117 is False Positive entities. This result is obtained when Word2vec(Mikolov *et al.*, 2013a) is used. There are other word embeddings models which were also used like GloVe(Pennington *et al.*, 2014) and Fasttext(Joulin *et al.*, 2016). In case of Glove word embeddings 29% and in case of Fasttext 31% precision is obtained. Not only different embeddings affect precision but also dimension of embedding affects the precision, following image show how precision varies with embedding dimension.

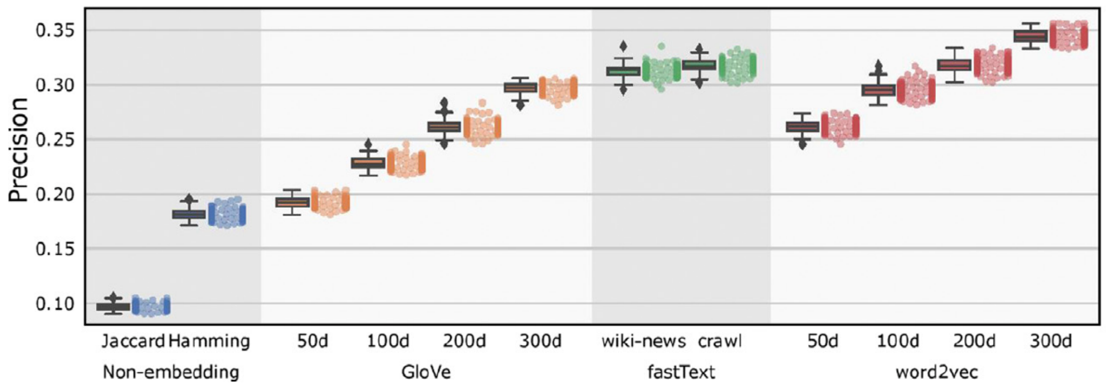

Figure 2.2: Precision varies with embedding dimension. highest precision of 0.34 is given by word2vec 300 dimension word vectors. Lowest precision given by Jaccard text similarity algorithm which is non-embedding based similarity algorithm. Same with hamming distance algorithm which is also non-embedding based similarity algorithm. Figure reprinted from (Youn *et al.*, 2020).

Apart from precision this Model also report 43.6% shorter path distance(hops) between predicted and actual class(2.91 vs. 5.16 when compared to other models). This means number of hops between actual class and predicted class for an entity is shorter in word embedding based model when compared to Jaccard and hamming distance based model.

As we have seen above LOVE model finds similarity of an entity with every class so hierarchical structure of an ontology is not being utilized, we shall see in next chapter utilization of hierarchical structure of an ontology. In LOVE model labels are used as unique identifier for each class and entity, there are total 2765 classes and 10897 entities in FoodOn ontology and using labels as unique identifier reduces this number to 2764 classes and 10865 entities which clearly indicated that there are duplicate labels. In Ontology Internationalized Resource Identifier(IRI) is used as unique identifier for classes and entities. Internationalized Resource Identifier is an internet protocol standard which builds on the Uniform Resource Identifier.

2.2 Summary

We have seen that LOVE model achieves precision of 34%, and this model considers every class to find the similarity with a candidate entity thus requires more time to complete population process. In the next chapter we shall discuss an efficient approach to populate FoodOn ontology which utilizes hierarchical structure of an ontology.

Chapter 3

Efficient Approach to Populate FoodOn Ontology

A semi-supervised approach for the FoodOn ontology population using word embedding. In this approach, we classify a new entity to existing ontology without comparing similarity with all the ontology classes, but a subset of classes is used to improve the time complexity. We are looking at the hierarchy to see it improves classification, as well as reduce time. We have seen LOVE Model in the previous chapter which considers all the classes to find scores with an entity and classify entity to class with the highest score. Apart from $score(c, e)$ as in LOVE model subtree score is also computed to traverse in ontology diagraph which is discussed later in the chapter. I have used fact that the child class inherits information from parent class hence parent can be represented as average of children's vectors. I have used IRI (Internationalized Resource Identifier) as a unique identifier for classes and entities in ontology where as labels are used as unique identifiers in LOVE model. I have used same sampling process as LOVE model where two entities are randomly chosen as seed entities. Class labels have some information about class as well as seed entities and child classes also have information about class, so we can use these information in some combination to obtain the vector for class. I have extended the work done by Jason Youn, Tarini Naravane, and Ilias Tagkopoulos(Youn *et al.*, 2020) to populate FoodOn ontology efficiently.

FoodOn class "Foodon product type" is the root class of all food products for humans. An acyclic digraph rooted at class "Foodon product type" is generated for this experiment, this acyclic graph is referred to as candidate ontology. A FoodOn class can have subclasses, and entities(food products). A class can have multiple parents also an entity can have multiple parents.

A seeded digraph is generated by randomly sampling one or two entities for each class as seeds from candidate ontology, and the rest of the entities used for ontology population, these entities are referred to as non-seed entities. For classes with more than two entities, only two entities are randomly sampled as seeds and for classes with exactly two entities only one entity is randomly sampled as seed, there are no seeds in classes without entities such classes are not considered during ontology population.

In word2vec embeddings, 103 classes out of 2765 have missing words in class labels these 103 classes have 189 total entities. 522 entities out of 10897 have missing words in entity labels. these are few examples of class labels with missing word

Class label	Missing word
food (barbequed)	barbequed
beverage (noncarbonated)	noncarbonated
seafood newburg (dish)	newburg
fruit preserves	preserves
chicken thigh (retorted)	retorted
sweetener (nonnutritive)	nonnutritive
mesogastropod	mesogastropod
neogastropod	neogastropod
fish, acipenseriform	acipenseriform

Table 3.1: Above table shows some of the words in class labels not in the vocabulary.

These are few examples of entity labels with missing words

Entity label	Missing word
beef hash (barbequed)	barbequed
shrimp (peeled, deveined, raw)	deveined
frozen nondairy dessert, chocolate, chilzert	chilzert
brazilnut (shell off)	brazilnut
popcorn (microwave with real butter and natural flavors)	flavors
splitnose rockfish (raw)	splitnose
chocolate eclair (frozen)	eclair
caramel candy with white centers	centers
coffee (canned, whitened)	whitened
brazilnut oil (food product)	brazilnut

Table 3.2: Above table shows some of the words in entity labels not in the vocabulary.

3.1 Terminology

- L_c is a label vector for class c. L_c equals to the average of word vectors in a class label with 15% more weightage to nouns since labels are noun phrases. Words not in vocabulary are ignored.
- L_e is a label vector for entity e. L_e equals to the average of word vectors in a entity label with 15% more weightage to nouns since labels are noun phrases. Words not in vocabulary are ignored.
- R_c is a representative vector for class c. R_c equals to the weighted average of L_c and average of seed entities label vectors in a class c i.e.

  $$R_c = \alpha L_c + (1-\alpha)\frac{1}{n}\sum L_e,\ \forall e \in c$$

  where $\alpha \in (0,1)$, and n is the number of seed entities in a class c. R_c is used for predicting entity to class.
- S_c is a subdigraph vector rooted at class c. S_c equals to the weighted average of R_c and average of child subdigraph vectors of class c i.e.

  $$S_c = \beta R_c + (1-\beta)\frac{1}{m}\sum S_{c_i},\ \forall c_i \in c$$

  where $\beta \in (0,1)$, and c_i is child class of c, m is the number of child classes in c. For leaf classes $S_c = \beta R_c$. S_c is used for deciding whether to explore child or not.

- RS_c is a subdigraph representative vector rooted at class c. RS_c equals to the average of class representative vectors of classes reachable from c i.e.

$$RS_c = \frac{1}{k}\sum R_{c_i}$$

  where c_i is a class which is reachable from c, including c itself, and k is number of such classes. RS_c is used for deciding whether to explore child or not.

- $Score(c, e) = sim\left(\vec{R_c}, \vec{L_e}\right)$, where $sim(.)$ is a cosine similarity.

Consider following example, a, b, c are classes. e_1, e_2, e_3, e_4 are seed entities. Let L_a, L_b, L_c are label vectors for classes and $L_{e_1}, L_{e_2}, L_{e_3}, L_{e_4}$ are label vectors for seed entities.

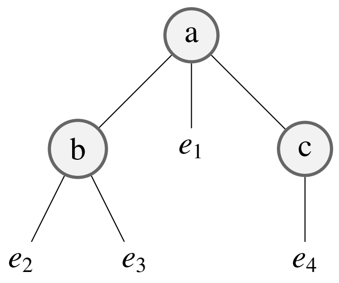

Figure 3.1: Consider this ontology for illustration.

Class b	$R_b = \alpha L_b + (1-\alpha)\frac{L_{e_2}+L_{e_3}}{2}$	$S_b = \beta R_b$	$RS_b = R_b$
Class c	$R_c = \alpha L_c + (1-\alpha)L_{e_4}$	$S_c = \beta R_c$	$RS_c = R_c$
Class a	$R_a = \alpha L_a + (1-\alpha)L_{e_1}$	$S_a = \beta R_a + (1-\beta)\frac{S_b+S_c}{2}$	$RS_a = \frac{R_a+R_b+R_c}{3}$

Table 3.3: Vector computation is shown for every class in Figure 3.1.

3.2 Mapping Non-seeds

Our task is to populate seeded digraph with non-seed entities. We take one non-seed entity say e at a time then traverse in a seeded digraph and try to find a class that has a high score with e, and assign e to the same class.

For this task, we start from the root of the seeded digraph and find similarity with all its immediate child subdigraph i.e. $\cos\theta_i$ between L_e and $RS_{c_i}, \forall c_i \in c$, where c_i is the immediate child class of class c. We traverse in every subdigraph rooted at child class c_i if $\cos\theta_i + bias$ is higher than that of class c for some $bias \in (0, 1)$ and repeat the steps for subdigraph rooted at c_i, meanwhile, we also compute $Score(c, e)$ for each class visited during traversal with e except for the classes without entities. If for every RS_{c_i}, $cos\theta_i + bias$ is less than that of class c then we backtrack since no child class is promising higher similarity. When traversal is completed we take the highest score and assign e to the same class.

Root class “Foodon product type” has 12 child classes eg. “Food product by Organism”, “Food product by quality”, “Food product by process”, “Food product by meal type”, etc. These classes are high-level classes hence for some of the non-seed entities similarity with these classes was less than that of the root class, so it could not traverse the correct subdigraph due to lower similarity, hence all of these 12 classes were considered for traversal separately.

Predicting the target class $\bar{c}$ to which candidate entity e mapped to can be formulated as follows:

$$\bar{c} = \arg\max_{c} Score(c, e)$$

where, c is visited during traversal.

Parameters of the algorithm:

- root: class to explore.
- entity: non-seed entity to be mapped.
- max_path_score: the maximum score of any class in the path from foodon_root to root.

“foodon_root” and “bias” are global constants. foodon_root is the root class of seeded digraph i.e. “Foodon Product Type”, and bias is set to 0.08 experimentally.

```
Effect: : entity.score is the maximum score with any class visited during
          traversal and entity.predicted_class is the same class.

Traverse-Digraph(root, entity, max_path_score):
if root.was_visited then
    return
end

score = cosine_similarity(root.$R_c$, entity.$L_e$)
if score > entity.score and count(root.seed_entities) > 0 then
    entity.score = score
    entity.predicted_class = root
end

max_path_score = max(class_path_score, max_path_score)
// class_path_score = cosine_similarity(root.$RS_c$, entity.$L_e$)

for child in root.children do
    child_path_score = cosine_similarity(child.$RS_c$, entity.$L_e$)
    if child_path_score + bias $\geq$ max_path_score or root == foodon_root then
        Traverse-Digraph(child, entity, max_path_score)
    end
end

root.was_visited = true
```

Algorithm 1: Traverse-Digraph. For populating seeded digraph, Traverse-Digraph(foodon_root, non-seed, 0) is called. Initial value for attributes: class.was_visited is false, entity.score is 0, entity.predicted_class is None. Time taken for this alsorithm is approx 5 *minutes*, whereas LOVE model takes approx 45 *minutes*.

The above method uses R_c vectors for prediction i.e. highest $Score(c, e)$ is used for predicting class for non-seed entities. RS_c vectors for traversal i.e. whether we should consider child subdigraph for traversal.

I have also experimented with S_c vectors for traversal instead of RS_c vectors and found lower precision. Similarly S_c, RS_c vectors were also considered instead of R_c

vectors for score calculation but precision was very low.

For testing the correctness of algorithm I have use inbuilt python 3.9 debugger to traverse the path followed by entity during population and verified the values of data structures on console for entire path.

3.3 Experimental Results

There are a total of 2765 classes and 10,897 entities in FoodOn ontology. 3474 entities randomly generated as seeds and 7423 as candidate entities. After populating seeded digraph by non-seed entities we compare each class in a seeded digraph with the same class in candidate ontology and count the number of non-seed entities populated correctly as TP(True Positive), non-seed entities populated incorrectly as FP(False Positive), and find the precision. Entities without word embeddings were ignored during precision calculation. *Bias* is taken because for some non-seed entities similarity with the child is less than that of the parent by a very small value.

Prediction	Traversal	Avg number of nodes visited	Bias	Precision(%)
Rc	Sc	537	.05	31
Rc	Sc	1127	.1	33
Rc	Sc	188	0	27
Rc	RS_c	30	0	20
Rc	RS_c	154	.05	32.9
Rc	RS_c	**300**	.08	**35.6**
Rc	-	2765(total classes)	-	35.9

Table 3.4: The table shows precision obtained for various combinations of bias and vectors used for prediction and traversal. $\alpha = 0.4$ and $\beta = 0.6$. Precision obtained by Jason Youn (Youn *et al.*, 2020) was $\approx 34\%$.

The **Traverse-Digraph** algorithm gives 35.6% precision and Jason Youn's method (Youn *et al.*, 2020) gives 34% precision. Jason Youn's method uses R_c with $\alpha = 0.2$ for prediction and visits all the nodes.

I have reviewed Jason Youn's implementation on GitHub and found a potential bug in the code. Jason Youn included approximately **380 seeds in non-seeds**. After excluding these seeds from non-seeds **precision falls to 31%**. The reason for lower precision could be using the class label, and entity label as a unique identifier because some classes have the same label as entities. I have used IRI (Internationalized Resource Identifier) as a unique identifier for classes and entities in ontology.

Some of the common words, for example, “dried”, “fried”, “cooked”, “sliced”, “diced”, “food”, “product”, “processed”, etc. are frequently occurring in entities and classes labels, I have removed these word from labels and achieved $\approx$ 35.6% precision. These words are common across classes hence unable to represent some specific classes. Taking maximum score with and without removing common words does not increase precision because some correctly classified entities get misclassified and vice-versa.

I have utilized Stanford Parser(D and CD, 2003) to extract the most meaningful word i.e. head of the noun phrase in labels but precision did not increase. Usually, the rightmost noun in the noun phrase is considered as head which fails in entity labels like “crabmeat/JJ cooked/JJ vacuum/NN packed/VBD”, here “crabmeat” should be a noun(NN) but tagged as an adjective(JJ) by the parser. If the entity label was “vacuum/NN packed/VBD cooked/JJ crabmeat/NN” then the parser would have returned the correct head, this shows that naming could have been better. Extracting first noun in a phrase also does not help, for example, “hen/NN egg/NN white/JJ boiled/VBD.” would return an incorrect head.

I have passed information from parent to child during precomputation of vectors but precision did not increase because as we move down in the ontology classes become more specific and adding information from parent generalise it. I have used non-compositional phrase (Mikolov *et al.*, 2013b) detection from gensim (Rehurek and Sojka, 2010) and achieved 29% precision which indicates phrase vectors are not effective.

Following are some interesting examples where our method failed to predict class correctly.

- Candidate entity: crabmeat (processed)
  - Correct class: crabmeat, score: 0.58
  - Predicted class: processed cheese food product, score: 0.65
- Candidate entity: pineapple (immature, sliced, in brine)
  - Correct class: pineapple food product, score: 0.53

  - Predicted class: fruit (sliced), score: 0.66

- Candidate entity: fruit dry diced glazed
  - Correct class: fruit food product, score: 0.73
  - Predicted class: fruit (dried), score: 0.76

- Candidate entity: reggiano cheese (block)
  - Correct class: hard grating cheese food product, score: 0.62
  - Predicted class: asiago cheese, score: 0.74

- Candidate entity: japanese spider crab
  - Correct class: oregoniid family, score: 0.35
  - Predicted class: spider crab family, score: 0.85

- Candidate entity: hen egg tuben
  - Correct class: prepared hen egg product, score: 0.80
  - Predicted class: hen egg liquid, score: 0.84

3.3.1 Holdout Method

Increasing the number of seeds for each class increases the precision. In a class having n entities, I have considered $n-1$ entities as seeds and one entity as candidate entity and achieved 48% precision. For 80% seed entities in each class precision is 44%. Seed entities represents the class and increasing the number of seeds makes class representation vectors more accurate.

No. of seeds	Precision
99%	48%
90%	46%
80%	44%

Table 3.5: Decreasing the number of seeds decreases the precision. Maximum precision achieved is 48%.

Following are some of the interesting examples where our method failed and even human classifier can fail.

- Candidate entity: cheddar cheese filled
  - Correct class: cow milk cheese analog, score: 0.73
  - Predicted class: cheddar cheese, score: 0.91
- Candidate entity: cheese food processed
  - Correct class: cow milk processed cheese product, score: 0.85
  - Predicted class: processed cheese food product, score: 0.86
- Candidate entity: soup mix dehydrated chicken
  - Correct class: soup mix product, score: 0.83
  - Predicted class: chicken soup mix, score: 0.93
- Candidate entity: grapefruit canned
  - Correct class: grapefruit food product, score: 0.76
  - Predicted class: grapefruit juice, score: 0.80
- Candidate entity: lobster raw quick frozen
  - Correct class: lobster food product, score: 0.79
  - Predicted class: lobster frozen, score: 0.89

3.3.2 BERT Embeddings

BERT, or Bidirectional Encoder Representations from Transformers, is a new method of pre-training language representations which obtains state-of-the-art results on a wide array of Natural Language Processing (NLP) tasks.

I have experimented with BERT(Devlin *et al.*, 2019) embeddings instead of word2vec embeddings and achieved 3% precision only. BERT provides contextual embedding for each word in sentence hence solves problem of polysemy but due to small vocabulary size precision is very low. There are total 4470 unique words in ontology labels out of which word2vec vocabulary has 4036 common words and BERT vocabulary has only 1565 common words. BERT has small vocabulary size of $\sim 30K$. In word2vec words which are not in vocabulary are ignored but BERT breaks such words into sub-words until sub-words are present in vocabulary eg. let “careful” is not in vocabulary then it is broken into “care”, “ful”, let “ful” is not in vocabulary then it is again broken into sub-words “f”, “u”, “l”.

I have used BERT model from Tensorflow hub It uses $L = 12$ hidden layers, a hidden size of $H = 768$, and $A = 12$ attention heads. This model has been pre-trained for English on the Wikipedia and BooksCorpus. Model url is: “https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4”, and text preprocessor url is:“https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3”

3.3.3 Fine-tuning BERT

In this experiment entity label vectors is obtained from BERT and same is supplied to feed forward classification neural network shown in 3.2 as input. Output of the model is predicted class index. Dimension of input layer is 768 which is BERT output dimension. Dimension of Dense layer is 2048 and dimension of softmax layer is $\approx$ 2429 i.e. number of classes in training data.

There is total 10857 entity class pair in an ontology. Classes are indexed, for a given entity label model is predicting the class index hence class label vectors are not utilized by model. 70% data is used for training and rest 30% for testing. This model achieved 15% precision when runs for 500 epochs, training it for more epochs does not improve precision however training for fewer epochs certainly affects precision.

While training the model BERT embeddings are not trained since embeddings are extracted from BERT then supplied to this model as input whereas weights in Dense Layer and Softmax Layer are tuned.

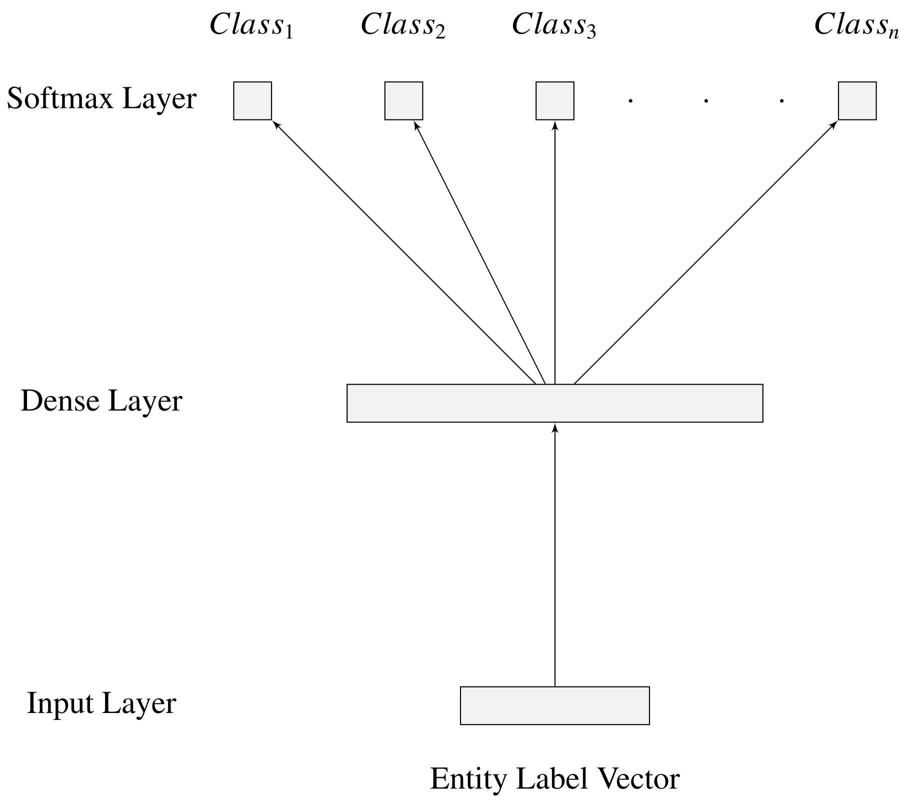

Figure 3.2: Fine-tuning BERT model for entity classification. Dimension of input layer is 768 which is BERT output dimension. Dimension of Dense layer is 2048 and dimension of softmax layer is $\approx$ 2429 i.e. number of classes in training data.

3.4 Summary

As we have seen various methods/techniques to boost the precision but we have failed. Until now class/entity labels were preprocessed and comma or parentheses were removed if any. But I have observed some patterns in naming convention of classes and entities which will be used to extract the head in next chapter.

Chapter 4

Rule-Based Parsing

In this chapter we discuss rule-based parsing methods for FoodOn ontology class/entity labels in order to extract the head of noun phrase.

I designed a rule-based parser to extract the head of noun phrase from class/entity labels. This parser is suitable for this particular problem since it is designed by analysing the class/entity labels. If label does not fall under any rule then no head is chosen. There are total 2765 class labels and 10897 entity labels out of which 24% class labels were selected to extract head and 54% entity labels were selected to extract head. There are nine rules given below with tables to demonstrate the accuracy of head extraction.

Extracted head in a class/entity labels is given 30% more weightage and rest of the nouns are given only 10% more weightage empirically. By using rule-based parsing we get precision of 36.5%. There are two types of phrases based on weightage of head and modifier, *heady* or *mody* (Kalouli *et al.*, 2019). *heady* phrases are in which head of noun phrase has higher weightage than modifier and *mody* being in which modifier of noun phrase has higher weightage than head (Kalouli *et al.*, 2019). Hence increasing the weightage of head only does not increase precision always.

The rules are as follows.

- Noun word before open parenthesis is the head. Total 5350 entity labels falls under this rule.

Entity label	Predicted Head	Actual class
broad bean (whole, dried)	bean	bean (whole, dried)
cowpea (pulse) food product	cowpea	bean food product
sausage (cooked smoked)	sausage	sausage (smoked)
goat milk cheese (unpasteurized)	cheese	goat milk cheese food product
hen egg (whole fried)	egg	prepared hen egg product

Table 4.1: If actual and predicted class are different then both are mentioned.

- Word on either side of "or" are head words. Total 41 entity labels falls under this rule.

Entity label	Predicted Head	Actual class/Predicted class
vegetable oil or fat for animal feed	oil, fat	plant fat or oil refined food product/ food (fat or oil coated)
pasteurized process cheese food with fruit or vegetable	fruit, vegetable	pasteurized process cheese food product
egg roll or bun	roll, bun	wheat roll or bun
wheat and corn macaroni or noodle product	macaroni, noodle	macaroni or noodle food product
product containing gravy or sauce	gravy, sauce	gravy or sauce

Table 4.2: If actual and predicted class are different then both are mentioned.

- Noun word after "based" in the label is the head. Total 42 entity labels falls under this rule.

Entity label	Predicted Head	Actual class/Predicted class
tomato-based hot sauce	sauce	condiment sauce/ tomato sauce
corn-based snack food	snack	field corn snack food product/ snack food
lemon breezer wine-based beverage	beverage	grape based alcoholic beverage/ wine or wine-like food product
dairy-based dessert	dessert	dairy dessert food product
fruit-based dessert energy reduced	dessert	fruit dessert food product

Table 4.3: If actual and predicted class are different then both are mentioned.

- Noun word before "packed in" is the head word. Total 6 entity labels falls under this rule.

Entity label	Predicted Head	Actual class/Predicted class
bonito packed in water canned	bonito	bonito food product
skipjack tuna packed in water	tuna	skipjack tuna food product
tuna packed in brine	tuna	tuna food product
fruit packed in extra heavy syrup	fruit	fruit food product/ fruit syrup

Table 4.4: If actual and predicted class are different then both are mentioned.

- Noun word before "with" is the head. Total 143 entity labels falls under this rule.

Entity label	Predicted Head	Actual class/Predicted class
butter with garlic	butter	cow milk butter food product
mushroom sauce with sliced mushrooms	sauce	vegetable based gravy or sauce food product/ mushroom vegetable food product
chop suey vegetables with imitation chicken	vegetables	multi-component vegetable product/ chop suey
cheese spread with meat flavoring	cheese	cow milk processed cheese product
cottage cheese with cream dressing	cheese	cottage cheese

Table 4.5: If actual and predicted class are different then both are mentioned.

- Noun word before "in" is the head word. Total 84 entity labels falls under this rule.

Entity label	Predicted Head	Actual class[Predicted class]
anchovy in olive oil	anchovy	anchovy food product
caper in vinegar	caper	caper food product
mushroom in buttered sauce	mushroom	mushroom vegetable food product
maraschino cherry in sugar syrup	cherry	cherry in syrup
chicken and vegetables in seasoned gravy	vegetables	chicken meat food product/ wheat based gravy or sauce food product

Table 4.6: If actual and predicted class are different then both are mentioned.

- Noun word before "made from" is the head. Total 18 entity labels falls under this rule.

Entity label	Predicted Head	Actual class/Predicted class
vinegar made from dried apple cores and skins	vinegar	apple vinegar food product
tomato juice made from concentrate	juice	tomato juice food product
fish paste made from salmon and other fish	paste	pacific salmon food product/ fish (dried)
bread made from milk, butter and egg	bread	bread food product
chili bean made from soy protein	bean	soybean substance/ soy protein

Table 4.7: If actual and predicted class are different then both are mentioned.

- Noun word before "for" is the head. Total 75 entity labels falls under this rule.

Entity label	Predicted Head	Actual class/Predicted class
fish for seafood salad	fish	fish product (unspecified species)/ multi-component seafood product
breading mix for frying	mix	wheat bread food product/ fat
salt block for livestock	block	salt product
fish byproduct for animal feed	byproduct	fish (whole or parts)
iodized water for livestock	water	food product for animal

Table 4.8: If actual and predicted class are different then both are mentioned.

- Noun word before first comma is the head. Total 120 entity labels falls under this rule.

Entity label	Predicted Head	Actual class/Predicted class
fish, spariform	fish	fish, bony
low alcoholic beverage, 3-5% alcohol	beverage	alcoholic beverage
orange, lemon and grapefruit marmalade	orange	citrus marmalade
vitamin supplement, prenatal	supplement	vitamin as food supplement food product
pancake syrup, artificially sweetened	syrup	food mix product/ flavoring syrup

Table 4.9: If actual and predicted class are different then both are mentioned.

4.1 Summary

We have seen experiments where random seed is allocated to each class, but now We will see how precision varies with varying seeds. In the next chapter I have proposed a method to find the near optimal seeds for each class.

Chapter 5

Discovering Optimal Seed entities

In this chapter I discuss a method to find near optimal seed entities. Let there are n entities in a class then $n - 20$ entities are chosen as seed entities and 20 entities as non-seeds. For each class 15 sets of $n - 20$ entities are created, and 15 times ontology population is done considering single set at a time. Out of 15 sets, one set is selected as optimal seed entities based on maximum number of non-seeds correctly predicted for same class. After selecting optimal seed entities for each class, ontology population is done one more time since all the classes have optimal seed entities. Classes which have less than 10 entities, all are chosen as seeds. Classes which have less than 30 entities only 10 are chosen as seeds. And classes which have more than 30 entities, 20 are chosen as seeds. Total seed entities are 9111(83%) and non-seeds are 1786(17%).

This method achieved precision of 49.6% which indicates that selection of seed entities greatly affects the precision. We have seen in Holdout method3.3.1 where $n - 1$ entities are selected as seed entities and precision obtained is 48% only so, selection of seed entities optimally can beat the holdout method.

The purpose of this experiment is to examine why model is not able to achieve good precision even after selecting $n - 20$ seeds optimally.

Following are examples of incorrect entity classification.

- Following entities belongs to class “meat food product”

Entity/Class Score	Predicted Class/Predicted Class Score
fish and meat blend (prepared) / 0.834	meat (prepared) / 0.858
intestine casings (edible) / 0.548	sheep meat food product / 0.621
seal meat (raw) / 0.906	meat (raw) / 0.912
suet (raw) / 0.686	food (raw) / 0.731
tankage (food product) / -0.046	spot / 0.108
armadillo meat (raw) / 0.869	poultry meat food product / 0.888
breakfast sausage (minimum cereal content of 6%) / 0.658	breakfast cereal / 0.830
marrow (ginger added) / 0.530	ginger food product / 0.724
alpaca meat food product / 0.811	meat (frozen) / 0.812
whole muscle meat cut / 0.797	meat (cut) / 0.863

Table 5.1: Table shows non-seed entity and corresponding predicted class with score. non-seed entities belongs to class “meat food product”

- Following entities belongs to class “cheese food product”

Entity/Class Score	Predicted Class/Predicted Class Score
cheese (soft, commercial) / 0.907	soft cheese food product / 0.938
cheese (mold-ripened, packed in hermetically sealed container) / 0.609	cured cheese food product / 0.649
cheese (maturing) / 0.791	semihard cheese product / 0.805
cheese (muenster, for manufacturing) / 0.879	semisoft cheese product / 0.880
cheese preparation (grated, unstandardized) / 0.833	grated cheese / 0.915
lingot cheese (with herbs and garlic) / 0.799	garlic food product / 0.838

Table 5.2: Table shows non-seed entity and corresponding predicted class with score. non-seed entities belongs to class “cheese food product”

- Following entities belongs to class “pork meat food product”

Entity/Class Score	Predicted Class/Predicted Class Score
pork (ground) / 0.902	processed pork meat food product / 0.910
pork (cut fresh) / 0.881	pork (raw) / 0.925
pork fatty tissue (partially defatted) / 0.643	animal tissue / 0.694
pork loin / 0.797	beef loin cut / 0.808
pork (canned) / 0.889	processed pork meat food product / 0.895
pork shoulder (raw) / 0.819	pork (raw) / 0.845
pork kidney / 0.829	kidney food product / 0.849
pork liver (frozen) / 0.833	liver food product / 0.887
pork (processed, spam-type) / 0.798	processed pork meat food product / 0.805

Table 5.3: Table shows non-seed entity and corresponding predicted class with score. non-seed entities belongs to class “pork meat food product”

- Following entities belongs to class “chicken meat food product”

Entity/Class Score	Predicted Class/Predicted Class Score
chicken and dumplings with vegetables (canned) / 0.795	mixed vegetables / 0.826
chicken neck (raw) / 0.832	chicken skin / 0.833
chicken spread / 0.846	food spread / 0.867
chicken cacciatore (canned) / 0.826	meat (canned) / 0.828
chicken neck (raw, ground) / 0.789	chicken leg / 0.818

Table 5.4: Table shows non-seed entity and corresponding predicted class with score. non-seed entities belongs to class “chicken meat food product”

- Following entities belongs to class “beef food product”

Entity/Class Score	Predicted Class/Predicted Class Score
beef chuck (roast cooked) / 0.670	roast beef / 0.837
wagyu steak / 0.716	beef steak (raw) / 0.861
standing rib roast / 0.157	roast beef / 0.491
beef (canned) / 0.902	meat product (canned) / 0.914
beef blood (raw) / 0.834	beef (raw) / 0.841
beef jerky / 0.841	meat (dried) / 0.847
chipped beef (smoked) / 0.843	beef (cooked) / 0.846
beef with gravy / 0.848	cattle broth or stock food product / 0.939
beef steak (fabricated) / 0.703	beef steak (raw) / 0.820
beef top round roast (cooked) / 0.723	roast beef / 0.947
beef ground round (cooked, freeze-dried) / 0.705	ground beef (cooked) / 0.885
beef chuck (raw) / 0.776	beef (raw) / 0.786

Table 5.5: Table shows non-seed entity and corresponding predicted class with score. non-seed entities belongs to class “beef food product”

5.1 Summary

As we have discussed even near optimal seeds(equivalent to human curated seeds) have failed to boost precision beyond 49.6%. Now we have to investigate why precision is not improving is it the quality of word vectors or the inherent nature of this problem to do so I have taken animal and plant ontology to populate in the next chapter.

Chapter 6

Animal Plant Ontology

In this chapter I shall discuss different ontology than previous work. I have collected animal names and animal classes from a-z-animals.com and plant names and its classes from herbs, shrubs, trees, creepers, climbers. There are total 9 animal classes with 558 animals and 5 plant classes with 417 plants. I created a class named “animals” and “plants” and put all the animal classes under “animals” and put all the plant classes under “plants” to create tree like structure as shown in Figure 6.2 and Figure 6.3, then created Animal Plant Ontology by merging “animals” class and “plants” class under “root” as shown below.

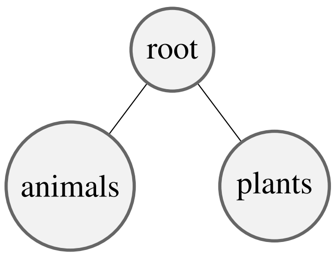

Figure 6.1: Animal Plant ontology.

The purpose of this experiment is to analyse the precision of ontology of different domain than Food. And also consider only single word entity labels to avoid the notion of noun phrases during label vector computation. L_c, L_e, R_c, S_c, and RS_c vectors are precomputed same as in 3.

I have extracted animal and plant dataset from same sources for training purpose. I have also queried Wikipedia for each animal/plant names to create wikipedia corpus for training. Preprocessed the corpus as follows: lower-case conversion, punctuation stripping, white-space stripping, numeric stripping, stop-word removal, and short words stripping.

For training, gensim (Rehurek and Sojka, 2010) implementation of the word2vec skip-gram model (Mikolov *et al.*, 2013a) is used. Default settings of the gensim word2vec model are used except for the following parameters: number of epochs of 250, window size of 5, and minimum count of 1, and vector dimension of 300.

I have computed word vectors for each class/entity same as described in 3 and run the ontology population algorithm 3 on this ontology with $\alpha = 0.1$ and each class/entity label as unique identifier, Table 6.1 show number of seeds verses precision. Maximum precision of 87% is obtained against 775 seeds.

No. of seeds	Precision
870	86%
775	87%
486	86%
92	77%
28	45%
0	39%

Table 6.1: Decreasing the number of seeds decreases the precision. Maximum precision achieved is 87%.

I have collected some of the incorrect predictions by model using 50% seed entities. Result is shown in Table 6.2. There are very few classes as compared to FoodOn ontology and also classes are mutually exclusive even though there are misclassifications which clearly indicates poor quality of word vectors. There are some misclassifications across animals and plants classes, which are completely different classes.

Entity label	Actual Class	Predicted class
puma	mammals, score: 0.09	malacostraca(crab), score: 0.12
tick	arachnida(mites), score: 0.36	clitellata(worms), score: 0.51
pumpkin	creepers, score: 0.34	herbs, score: 0.56
chicken	birds, score: 0.49	herbs, score: 0.51
cuscus	mammals, score: 0.15	creepers, score: 0.23
pyracantha	shrubs, score: 0.48	clitellata(worms), score: 0.54

Table 6.2: Last three rows shows incorrect prediction between animals and plants classes.

6.1 Animal Ontology

I have experimented with animal ontology to understand the impact of training data extracted from a-z-animals.com. I have taken only animal ontology as shown in Figure 6.2. Then I trained two set of word vectors one is trained on a-z-animals.com corpus and wikipedia corpus whereas other is trained on only wikipedia corpus. Then I run ontology population algorithm and found difference in precisions. First set of word vectors performed better than other since it was trained on both corpus which says that more training data improves quality of word vectors. Hence training word vectors on animal related corpus could increase the precision even further.

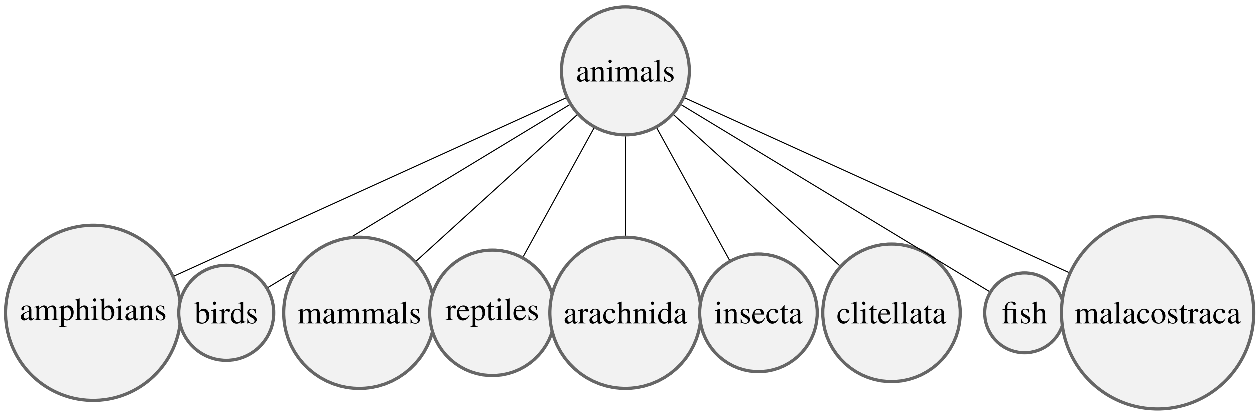

Figure 6.2: Animal ontology. There are total 9 animal classes with 558 animals.

No. of seeds	Precision	Precision(without animal corpus)
496	89%	84%
444	92%	85%
277	90%	83%
53	82%	71%
18	46%	42%
0	35%	34%

Table 6.3: Decreasing the number of seeds decreases the precision. Maximum precision achieved is 92%.

6.2 Plant Ontology

Similar to animal ontology I have run the ontology population algorithm for plant ontology just to see performance individually. Result is show in Table 6.4. Maximum precision of 86% is obtained which is adequate and it indicates that precision of animal plant ontology is not hampered by plant ontology or animal ontology.

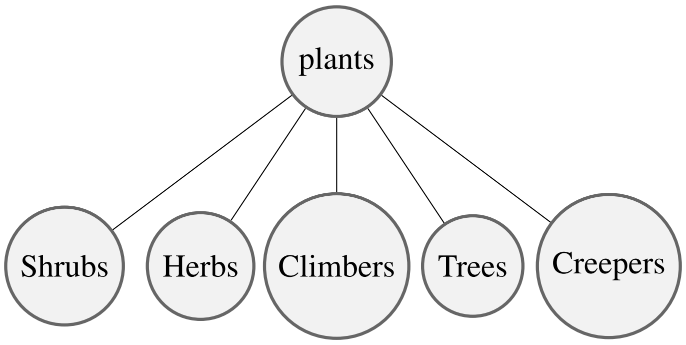

Figure 6.3: Plant ontology. There are total 4 plant classes with 417 plants.

No. of seeds	Precision
375	86%
333	85%
209	84%
39	76%
10	58%
0	56%

Table 6.4: Decreasing the number of seeds decreases the precision. Maximum precision achieved is 86%.

6.3 Summary

In this chapter we have discussed an ontology from different domain i.e. Animal Plant Ontology which achieved maximum precision of 87% at 775 seeds which is greater than that of FoodOn ontology. Also in Animal Ontology we have seen domain related data for training increases precision.

Chapter 7

Conclusions and Future Work

I demonstrated an efficient semi-supervised ontology population method using word embeddings our method is $\approx$ 88% faster with precision of $\approx$ 35.6% as compared to LOVE Model(precision $\approx$ 34%). All the phrases are not compositional, phrase vectors were not effective due to the unavailability of large datasets for training. Using non-compositional phrases gives $\approx$ 29% precision, whereas using compositional phrases gives $\approx$ 35.6% precision. I have also shown an inefficient way that achieved a precision of $\approx$ 35.9%. Considering $n-1$ seed entities achieves a precision of $\approx$ 48% whereas using $n-20$ seeds optimally yeilds $\approx$ 49.6% precision. That indicates that the quality of word vectors is poor. Many words in entity labels have no vectors which degrade the quality of label vectors that might be the reason for low precision. Our efficient method could be used in other domains where word embeddings based approach is prominent. Using rule-based parsing for head extraction 36.5% precision is obtained. Fine-tuning BERT model gives 15% precision where 30% entities were used as test data. Animal Plant Ontology populations shows that getting precision more than 49% is possible. Domain related data for training affects the precision as we have seen in case of Animal Ontology. In FoodOn Ontology some entities could not be even classified by humans.

Class/entity labels are noun phrases. Explore the literature of noun phrase embedding and applying the suitable approach to get quality phrase embedding shall be useful. Collecting food related data via web scraping for training might improve the quality of word vectors.

References

D, K., and CD, M., 2003, “Accurate unlexicalized parsing,” (In: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics- Volume 1. Sapporo: Association for Computational Linguistics).

Devlin, M., Chang, W., Lee, K., and Toutanova, K., 2019, “Bert:pre-training of deep bidirectional transformers for language understanding,” (In NAACL-HLT).

Jayawardana, V., Lakmal, D., de Silva, N., Perera, A., Sugathadasa, K., Ayesha, B., and Perera, M., 2017, “Semi-supervised instance population of an ontology using word vector embeddings,”

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T., 2016, “Bag of tricks for efficient text classification,” (arXiv Preprint arXiv1607.01759).

Kalouli, Aikaterini-Lida, Paiva, D., Valeria, Crouch, and Richard, 2019, “Composing noun phrase vector representations,” (84-95. 10.18653/v1/W19-4311).

Karadeniz, I., and Ozgur, A., 2019, “Linking entities through an ontology using word embeddings and syntactic re-ranking,” (BMC Bioinformatics).

Mikolov, T., Chen, K., Corrado, G., and Dean, J., 2013a, “Efficient estimation of word representations in vector space,” (Proceedings of the International Conference on Learning Representations).

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J., 2013b, “Distributed representations of words and phrases and their compositionality,” (In Advances in Neural Information Processing Systems).

Pennington, J., Socher, R., and Manning, C., 2014, “Glove: Global vectors for word representation,” (in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), Doha, Qatar (Stroudsburg, PA: Association for Computational Linguistics) 1532–1543).

Rehurek, R., and Sojka, P., 1993, “A translation approach to portable ontology specifications,” (Knowledge Acquisition).

Rehurek, R., and Sojka, P., 2010, “Software, framework for topic modeling with large corpora,” (LREC 2010 workshop on new challenges for NLP frameworks (ELRA), Paris, France).

Youn, J., Naravane, T., and Tagkopoulos, I., 2020, “Using word embeddings to learn a etter food ontology,” (Frontiers in Artificial Intelligence).

Acknowledgements

I am extremely grateful to my MTP guide **Prof. Abhiram Ranade** for helping me throughout the research project. The project meeting discussions have been very helpful in resolving doubts related to the Word Embeddings. Secondly, I would also like to thank my parents for their constant encouragement and support during the online semester.

Deepak Singh Baghel
IIT Bombay
22 June 2022