



INTERNATIONAL INSTITUTE OF
INFORMATION TECHNOLOGY

H Y D E R A B A D

Introduction to NLP

Hypernym Discovery

Final Report

Team 56

Ayush Mittal - 2021201030

Swarnali Dey - 2021201088

Aditya Gonnade - 2021202023

1. Project Description

1.1 Hypernym-Hyponym Introduction and Importance

A hypernym is a word that names a broad category that includes other words. In other words, hypernym can be considered as the base class of objects where each such object is called a hyponym. A hyponym is in a type-of / is-a relationship with its hypernym. For example, pigeon, crow, eagle, and seagull are all hyponyms of hypernym bird, and bird itself is a hyponym of hypernym animal.

Hypernym discovery aims to discover the set of hypernyms given a hyponym and proper corpus. Hypernymy relationship plays a critical role in language understanding because it enables generalization, which lies at the core of human cognition. It has been widely used in various NLP applications from word sense disambiguation to information retrieval, question answering and textual entailment.

1.2 Scope of the project

The project's scope is as follows - given a set of input terms (concept or entity) and a vocabulary, we need to retrieve a ranked list of candidate hypernyms (up to 15) for each input term.

We are implementing the following research paper to retrieve the list of 15 hypernyms.

“CRIM at SemEval-2018 Task 9: A Hybrid Approach to Hypernym Discovery”

Link: <https://aclanthology.org/S18-1116/>

In this research paper, there is a hybrid model which uses two approaches - a supervised projection learning approach and an unsupervised, pattern based approach. The results from both these approaches are combined to generate the final list of hypernyms. The word embeddings used in this approach are pretrained word2vec embeddings which are further trained with negative sampling on hyponym-hypernym pairs.

Since the number of hyponym-hypernym pairs in the dataset is less (1500 for English dataset, 500 for medical and music dataset), hearst patterns are used to generate more hyponym-hypernym pairs from the unlabeled corpus and added to the original dataset.

2. Dataset and Corpus

There are 5 subtasks, for every subtask we have an unlabeled corpus and dataset. Dataset have three sections -

1. Data - containing a list of queries (hyponyms) with their class as concept or entity.

2. Gold Section - containing some hypernyms for the queries (hyponyms) in the data section. Each line of the gold text file contains the set of hypernyms corresponding to the query (hyponym) given in the same line in the data file.
3. Vocabulary - containing all the words to consider for vocabulary for each corpus.

In the dataset, vocabulary is already given, all the queries will be from this vocabulary and the desired output is some fixed number (say 15) of hypernyms from the corpus. There is a gold dataset which contains a list of hypernyms of all the possible hyponyms, but the number of hypernyms present in the gold dataset is very less than desired. Thus, we have to generate more hypernyms for each such hyponym in the data file.

3. BaseLine Model

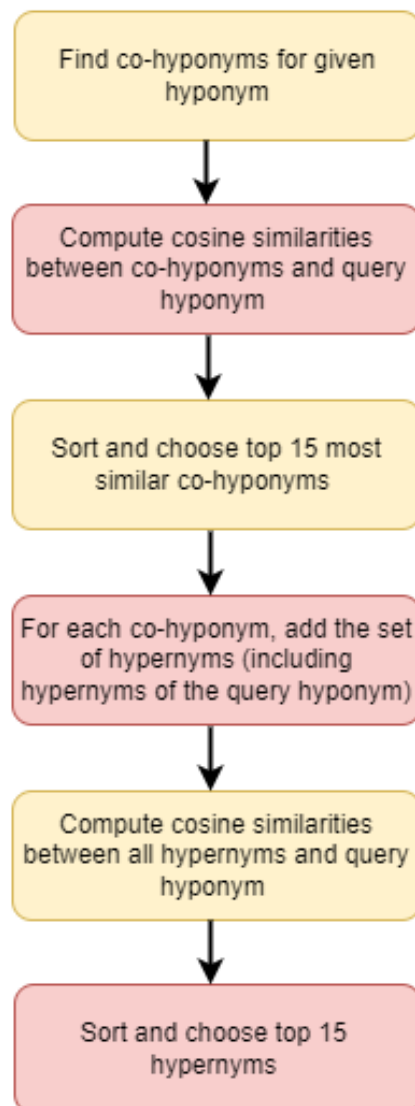
Our baseline model was implementation of the unsupervised learning approach where we used Glove embeddings for the words in our vocabulary and generate top 15 hypernyms for a given hyponym based on the hyponym-hypernym relations given in the original data and gold datasets alongwith those pairs extracted using Hearst patterns. The approach is discussed in detail in the next section.

3.1 Unsupervised Learning Approach

As mentioned, the dataset contains three main sections, one is the gold section, which contains a small list of hypernyms, and since this gold dataset contains insufficient number of hypernyms for some queries, we handle this by enriching the list of hypernyms using the following algorithm -

- Initially we had only a small list of hypernyms for all the words in the list, and our task is to get 15 closest hypernyms for a given query. So before finding the closest hypernyms we need to extend the list of the total number of hypernyms.
- We first construct two dictionaries - one containing a list of hypernyms for given hyponyms and another containing a list of hyponyms for given hypernyms.
- We look for co-hyponyms for the given query, and then pick the closest co-hyponyms. Co-hyponym is a word or phrase that shares the same hypernym as another word or phrase.
 - To find co-hyponyms, we make a list L of available hypernyms of the query.
 - Then make a set S of all the hyponyms of all the elements present in list L.
- For finding the closest co-hyponyms, we used the frequency and cosine similarity between the co-hyponym and query.
 - For a co-hyponym c, frequency is the count of hypernyms which c has in common with the query.
 - For finding similarity, we are using the product of frequency and cosine similarity between the embeddings of the query and the co-hyponym.
- We now keep the 15 most similar co-hyponyms and discard the rest.
- Then we make a set H of hypernyms, for all the closest 15 co-hyponyms, and we insert all their hypernyms in this set.

- Finally we have an extended list of hypernyms, and now the final task is to find the 15 closest hypernyms for the given query. For this we first computed the cosine similarity score between the embeddings of the query and the extended list of hypernyms, then computed the product of the frequency of the hypernym and its cosine similarity score, ranked all the candidate hypernyms using this score and picked top 15 from this list of ranked hypernyms.
 - This time frequency for hypernym is the count of co-hyponyms which are having this hypernym.



3.2 Results

Although we have used pretrained glove embedding, we are getting good results. Some of the results are -

Query	Top 15 predicted Hypernyms
abhorrence	Hatred disgust hate distaste hostility offensive activity dislike opposition rejection oppositeness social rejection aversion silent treatment rebuff hindrance
cumulus	natural phenomenon observable phenomenon atmospheric phenomenon cloud dust air pollutant air quality pollution natural environment air pollution particulate matter endangerment disaster natural disaster
sodium nitrite	Bond chemical bond chemical series chemical group inorganic compound paper copy building block molecular entity carbon group coated paper non-metal main group nonmetal main group element
silver	Riches native metal historic house tableware noble metal base metal badge of honour grey metallic element runner-up wealth atom greyness grayness molecular entity

In the above table we can clearly see that nearly all the hypernyms are correct for the given query. The whole output file is uploaded in a google drive folder, and link of the folder is -

https://drive.google.com/drive/folders/1wx3cyUffR_h0XM2Cof8untoFUGTkqWHQ?usp=sharing

This folder contains the data, gold and output file.

4. CRIM's Hybrid Approach

We will be improving our baseline model in the final submission. The steps we would be following for improving the solution are mentioned below -

4.1 Extracting hyponym-hypernym pairs using Hearst patterns from unlabeled corpus

The Hearst patterns are linguistic patterns used to recognize hyponyms (specific terms) and hypernyms (general terms) in text. For example, in the sentence "Dogs, cats, and birds are examples of pets," "pets" is a hypernym for "dogs," "cats," and "birds," while "dogs," "cats,"

and "birds" are hyponyms of "pets." The goal of the code is to find such hyponyms and hypernyms in the UMBC corpus.

The HearstPatterns class defines the patterns used to recognize hyponyms and hypernyms and provides two methods for finding hyponyms in text: chunk and find_hyponyms.

The init method initializes the class by defining a list of adjective stopwords, the Hearst patterns to use, and the spacy natural language processing object to use for text processing. The patterns are defined as tuples of regular expressions and string literals that indicate whether the first or last noun phrase in the pattern is the general term (hypernym) or the specific term (hyponym).

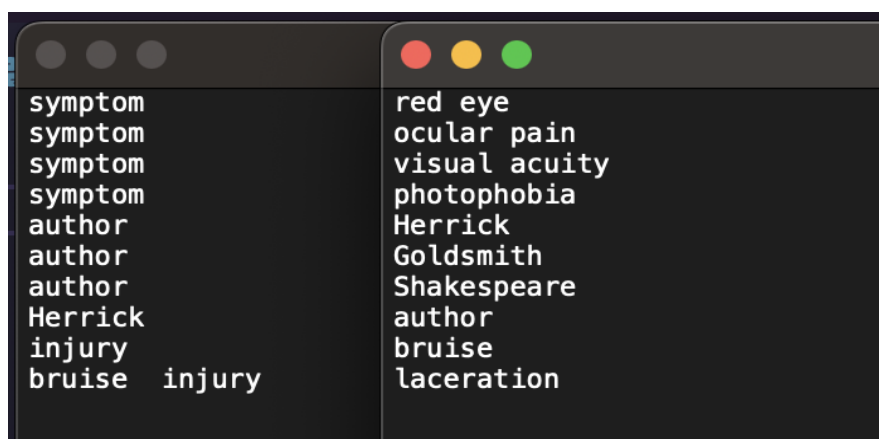
The chunk method takes raw text as input and returns a list of tagged noun phrases in the text. It first processes the text using spacy to tokenize and lemmatize the text and extract noun phrases. It then replaces each noun phrase in the text with a tagged version of the phrase, where the tag is NP_ followed by the lemmatized words of the phrase. The replace_arr list is used to generate the tag by removing stopwords, punctuation, and other non-alphanumeric characters from the phrase.

The find_hyponyms method takes raw text as input, processes it using chunk to extract tagged noun phrases, and then applies the Hearst patterns to the tagged phrases to find hyponyms and hypernyms. For each sentence in the tagged text, it checks each Hearst pattern against the sentence to find matches. If a match is found, it extracts the general and specific terms from the match using regular expressions and appends them to the hyponyms list. The method returns the hyponyms list, which contains tuples of (specific-term, general-term) pairs.

For input of few sentences we got the output as show in below image:

Example :

1. Forty-four percent of patients with uveitis had one or more identifiable signs or symptoms, such as red eye, ocular pain, visual acuity, or photophobia, in order of decreasing frequency.
2. There are works by such authors as Herrick, Goldsmith, and Shakespeare.
3. There were bruises, lacerations, or other injuries were not prevalent.
4. Many countries, especially France, England and Spain also enjoy toast.



symptom	red eye
symptom	ocular pain
symptom	visual acuity
symptom	photophobia
author	Herrick
author	Goldsmith
author	Shakespeare
Herrick	author
injury	bruise
bruise	laceration

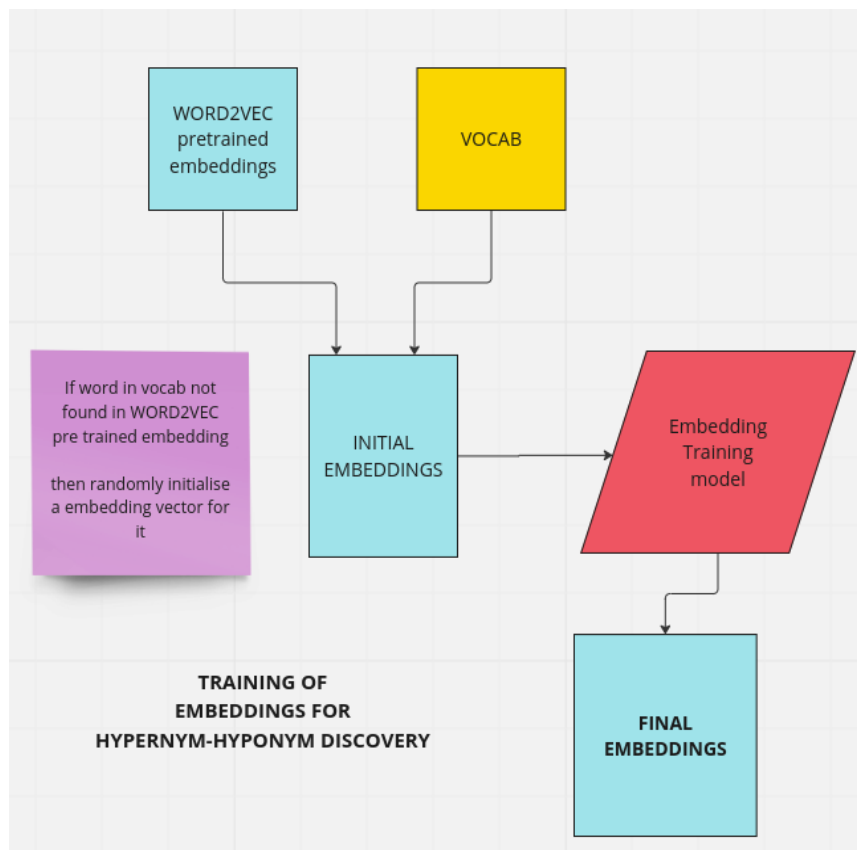
4.2 Pre-processing dataset

The corpus for all the sub-tasks are pre-tokenized so we preprocess the corpus in the following steps -

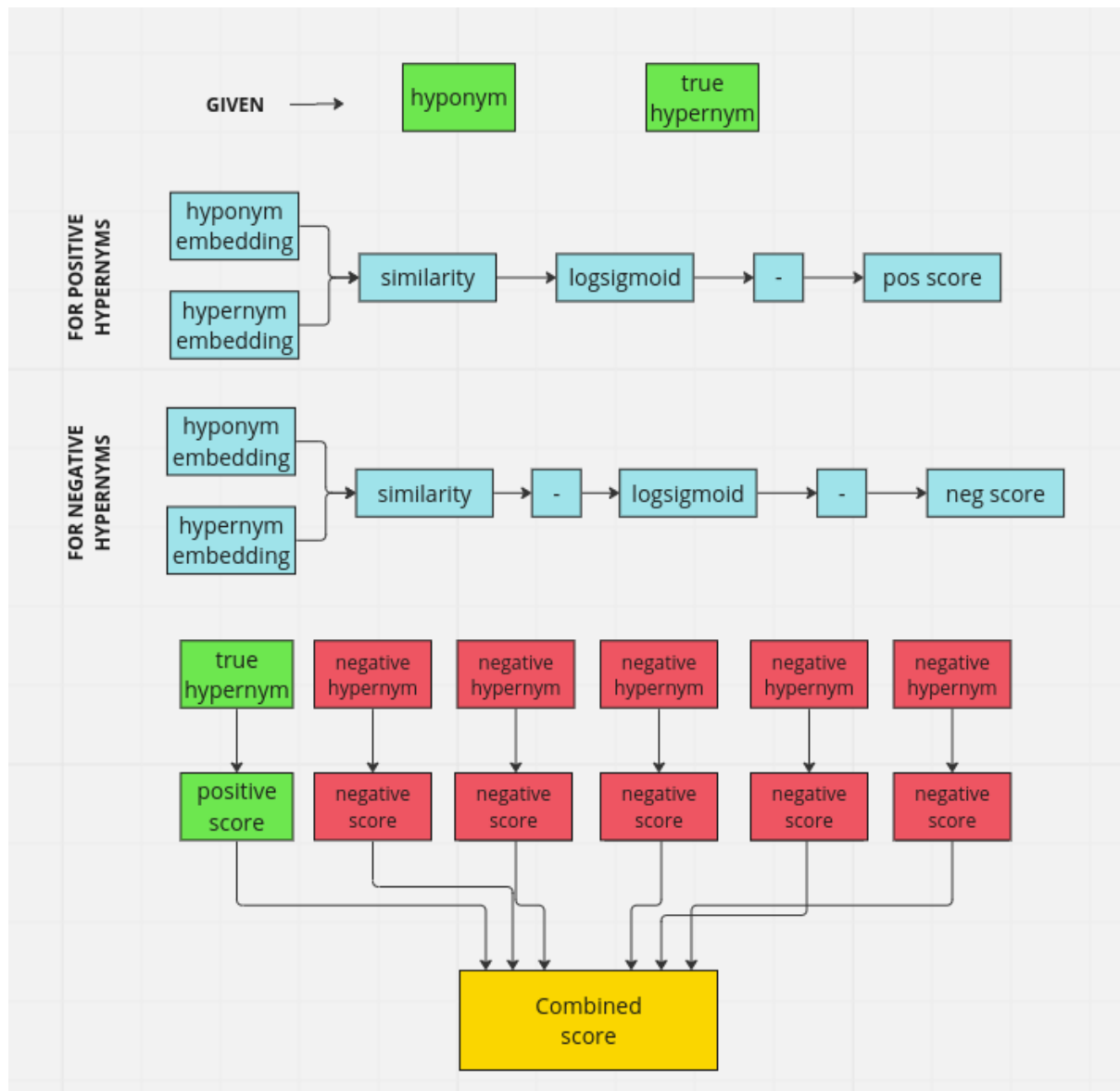
- ❖ Split into sentences and tokenize the sentences to store hyponyms and hypernyms lists.
- ❖ Converting all words to lowercase
- ❖ Replace all multi-word terms found in vocabulary by a single token (join them using underscores).
- ❖ Add the hyponyms and hypernyms not in the vocabulary to our vocabulary and use them to further train our Word2vec embeddings.

4.3 Training Word2Vec embeddings

The embeddings used are pretrained Word2Vec embeddings which are further trained according to hyponym-hypernym pairs. Each word in the vocabulary is assigned its corresponding pretrained word2vec embedding and if for a word pretrained embedding is not present (like for 2 or 3 word phrases and proper nouns like people's names or locations), we assign a random embedding and then train it with the rest of the word embeddings.



In the embedding training model, for each hyponym, we extract the list of positive hypernyms and for each such positive hypernym, we generate 5 negative hypernyms. We then calculate the similarities between (query hyponym, positive hypernym) and (query hyponym, negative hypernyms) and then pass through a logsigmoid layer. The negative scores for each negative hypernym is added to calculate the resultant negative score. Thus, now we have a single positive score and a single negative score. We finally calculate the mean of both these scores and treat it as the loss. This loss is then backpropagated to train the embeddings.



To check how our trained word2vec embeddings are performing compared to pretrained Glove embeddings, we generate top 15 hypernyms using the unsupervised model individually with trained word2vec and Glove embeddings. Results are as shown below.

Hyponym: tropical_storm

Given hypernyms:

['atmosphere', 'windstorm', 'violent_storm', 'air_current', 'atmospheric_state', 'density', 'current_of_air', 'storm_damage', 'atmospheric_phenomenon', 'storm', 'cyclone', 'natural_phenomenon', 'tempest', 'wind']

Hypernyms generated using Glove embeddings:

['catastrophe', 'trouble', 'body_part', 'impinging', 'denseness', 'misfortune', 'risk', 'difference_of_opinion', 'ideal', 'windstorm', 'physiological_reaction', 'variable', 'musical_work', 'inclination', 'hostility']

Hypernyms generated using custom trained Word2Vec embeddings:

['natural_phenomenon', 'phenomenon', 'physical_phenomenon', 'weather_condition', 'atmospheric_phenomenon', 'weather', 'atmospheric_condition', 'violent_storm', 'current_of_air', 'storm', 'wind', 'air_current', 'windstorm', 'hurricane', 'atmosphere']

We can see from the above results that with Glove embeddings some random and unrelated hypernyms are produced for the query hyponym “pollution”, while with our custom trained word2vec embeddings, much better hypernyms are being produced.

4.4 Supervised Approach

Before going into the details, let's have a look at how this model will be working -

For the given query we will consider all the hypernyms, then will find out the likelihood score of hypernyms and query with the help of the model. Then sort these hypernyms according to the likelihood score and will provide the top hypernyms as the output.

So basically the model is used for finding the likelihood score for the given pair of query (hyponym) and hypernym.

4.4.1 The Model

- For the given query and hypernym, firstly find out the embedding E_q and E_h respectively (word2vec embeddings), which we have stored in the above section.
- Then will find out k projections, for each projection, query embedding E_q is multiplied by square projection matrices.
 - E_q is of $d * 1$ dimensions
 - Each projection matrix is of $d * d$ dimensions
 - Initially these projection matrices are having random values.

$$P_i = (Q_i \cdot E_q)^T$$

- Then these projections are checked as to how close they are from the hypernym embedding.

$$s = P \cdot E_h$$

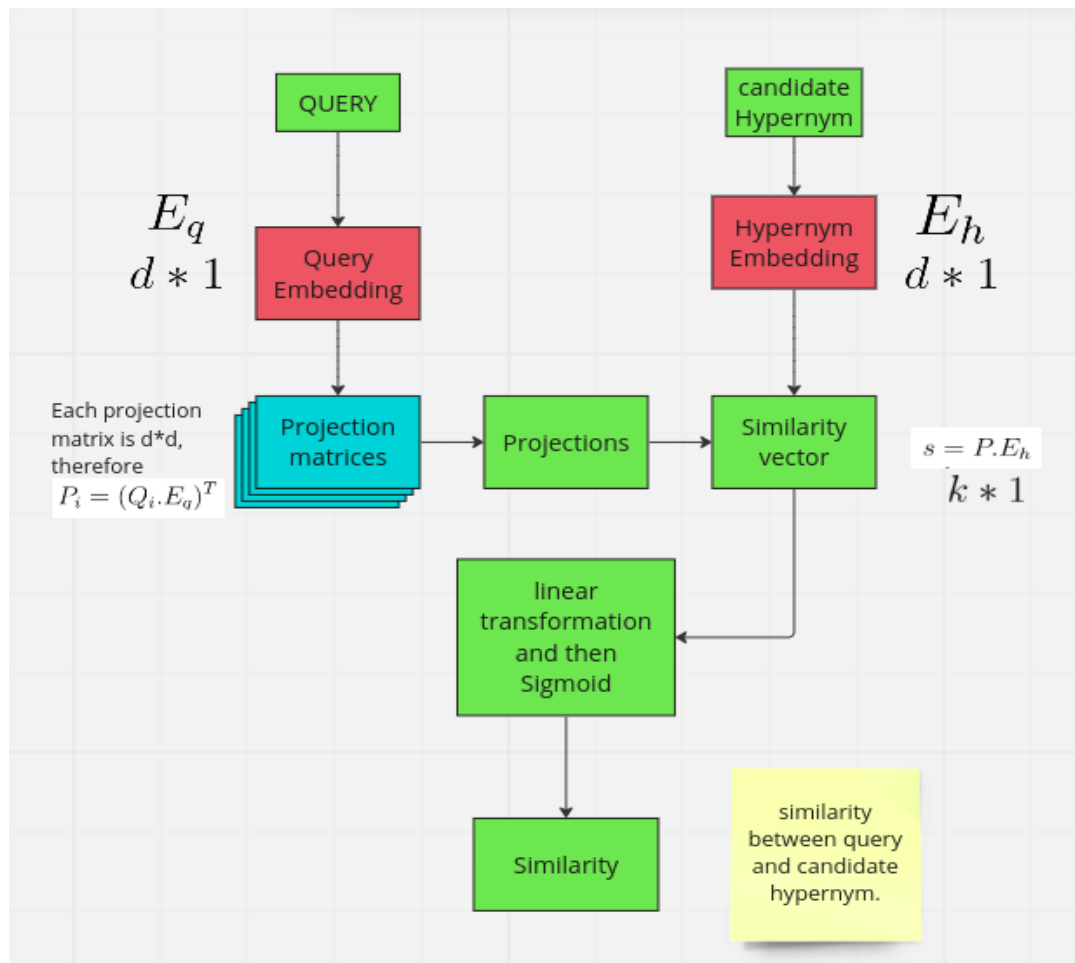
s will be a column vector ie of size $k * 1$

- Now this dot product is fed through an linear transformation and then through a sigmoid function. The output will be the similarity between hypernym and query.

$$Y = \sigma(W \cdot s + b)$$

Here W and b are parameters for affine transformation

If the given pair is true i.e. candidate hypernym is really a hypernym, then the likelihood should be close to 1, else close to 0.



4.4.2 Training of Model

We will be using negative sampling, that is, for each positive pair of hypernym-hyponym, we will train the model with negative samples also, so that our model can know which is actually a hypernym and hyponym pair and which is not. For every true pair, we will train 5 negative pairs - generating negative pairs is done by replacing hypernym with a word randomly chosen from the vocabulary.

As mentioned earlier, we will train the model such that it gives likelihood close to 1 for true pairs and close to 0 for negative pairs, and for this we will use binary cross entropy loss. The formula to be used is mentioned below -

$$H(q, h, t) = t * \log(y) + (1 - t) * \log(1 - y)$$

Here,

q is query

h is hypernym

y is likelihood given by model

t is 1 if true pair, else 0 in case of negative pair.

For the whole batch containing positive and negative pairs of hypernyms and hyponyms, we will sum the loss and to minimize the loss, we will use gradient descent, so the model is trained as desired.

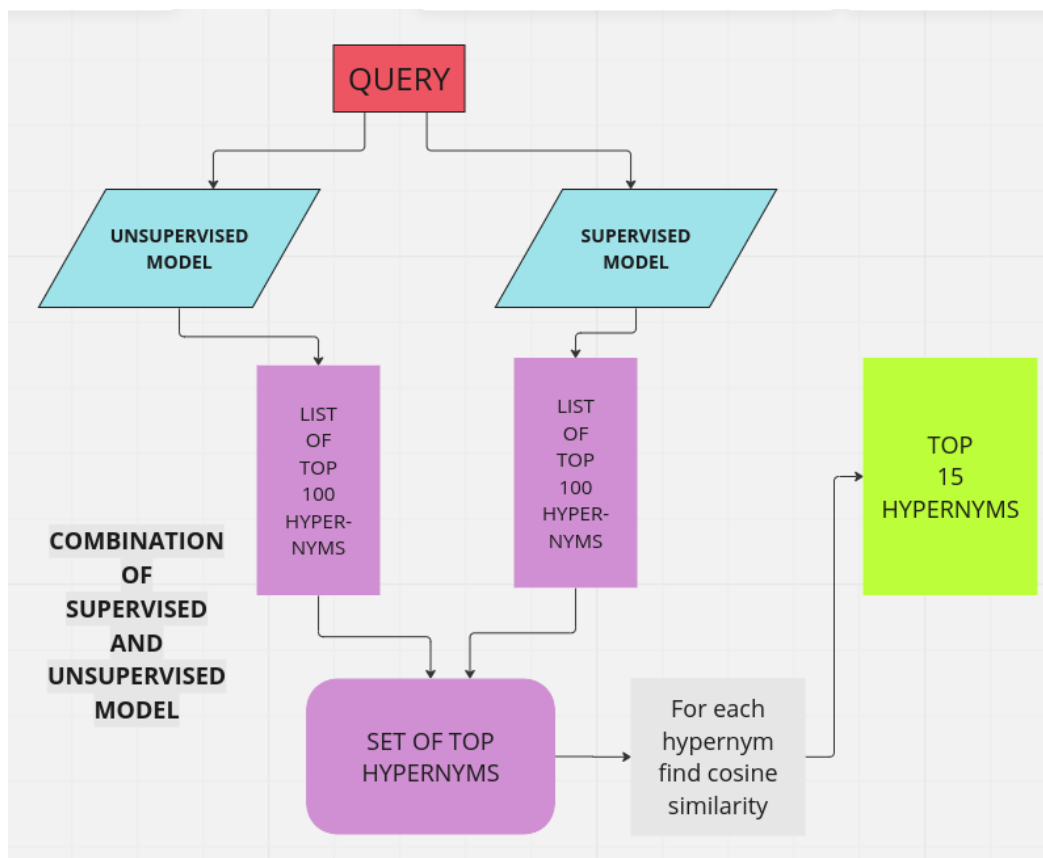
Some of the important points are -

- We will use a fixed number of projections i.e. we will keep the value of $k = 24$, not dynamic.
- As stated earlier, initially we will use randomly initialized projection matrices.
- Word embeddings are optimized during training.

4.5 The Final Hybrid model

The final model is the hybrid of the above mentioned supervised and unsupervised model. For the given query, the hybrid model will obtain 100 hypernyms each from the supervised and unsupervised model for the given query. For the unsupervised model, it may be possible that 100 hypernyms cannot be produced, then we take as many hypernyms that can be generated. Then we calculate the cosine similarity scores for each hypernym with the query hyponym. On the basis of these scores, we sort and rerank the hypernyms. The output will be the top 15 hypernyms from the final list.

The advantage of this hybrid model is, it will promote the hypernyms obtained in both supervised and unsupervised approach, and also favour those hypernyms which are strongly predicted from either supervised or unsupervised model.



5. Analysis

The unsupervised model works better compared to the supervised model and sometimes even better than the hybrid model. Some results are shown below.

```
Hyponym: pollution
Top 15 hypernyms from supervised model
['person', 'chief', 'locale', 'boss', 'leader', 'politician', 'transmission', 'movement', 'channel', 'show', 'computer', 'competitor', 'worker', 'painter', 'wind']

Top 15 hypernyms from unsupervised model
['report', 'hearsay', 'impurity', 'junk', 'waste_material', 'defaecation', 'uncleanness', 'shit', 'excretion', 'find', 'granular_material', 'garbage', 'refuse', 'waste_product', 'dog_shit']

Top 15 hypernyms from hybrid model
['report', 'hearsay', 'person', 'impurity', 'junk', 'waste_material', 'defaecation', 'uncleanness', 'shit', 'excretion', 'find', 'granular_material', 'garbage', 'refuse', 'chief']
```

The supervised learning approach does not give accurate results mainly because of lack of training data (hyponym-hypernym pairs). The Hearst pattern retrieval from the entire (48GB) was not possible. We extracted Hearst patterns from 1 file (size - 48MB) among 400+ files, but the hyponym-hypernym relations extracted were not of good quality.

Mean Average Precision Scores

Subtask	Unsupervised Approach	Hybrid Approach
English (1A)	51.65%	15.8%
Medical (2A)	50.45%	22.8%
Music (2B)	39.79%	13.29%

We can see that the precision score for unsupervised learning approach is much better than the hybrid approach for all subtasks.

GOOGLE DRIVE-

https://drive.google.com/drive/folders/1KwTvPf3Tj3PxS_Oq_V7UNoYT3YT_wjc?usp=sharing

GITHUB -

<https://github.com/AyushMittal1109/Hypernym-Hyponym-Discovery>