Report: Knowledge agnostic language modeling and Entity sense discovery

Advait Savant

January 2024

1 Introduction

Link to the GitHub repository

In data folder of the default master branch, the post-processed kalmified corpus is present in a csv file.

In the entity sense discovery folder, the entity sense data json file contains the data collection (words, assoicated sentences in different contexts) necessary for entity sense discovery.

In the entity sense discovery folder, the entity sense embeddings pickle file contain the sense emebbings for distinct senses of a concerned word.

Code is structured for scalability and modularity. A separation exists between the generation, management, updation and retrieval of data and methodology for algorithmic design in view of processing of data.

25.7 percent reduction is obtained in the vocabulary size post kalmification. Distinct entity senses discovered, and their corresponding sentences, seen to align with semantic notions.

2 Kalmified Corpus

The Kalmified corpus [1] is created for the ConLL-2012 ontonotes dataset. Processed 1,31,492 sentences in the training and validation datasets.

Appropriate entity spans are accounted for and tested for with BIO-tags. Pre-processing aspects like, 's after entities, are accounted for.

The following entity types are kalmified and exhaustively accounted for.
1. PERSON: People's names 2. NORP: Nationalities, religious,or political

groups, 3. FAC: Facilities, 4. ORG: Organizations, 5. GPE: Geopolitical entities, 6. LOC: Locations, 7. VEHICLE : locomotives, 8. DATE : Dates, years, timespans, 9. TIME : Times of day, hours, durations, periods of the day, 10. NUMBER : numbers, 11. PERCENT : percentages, 12. MONEY : Monetary values, financial amounts, 13. QUANTITY : Quantities, measurements, 14. ORDINAL : Ordinal numbers, 15. CARDINAL: Cardinal numbers, statistical figures 16. EVENT : Events, incidents, and notable occurrences 17. WORK OF ART : literature, media titles 18. LAW : Legal documents, laws, legal terms 19. LANGUAGE : Languages, linguistic groups 20. MISC : miscellaneous

Methods for the expansion and management of the corpus are set up. As our ontology get's more and more detailed and refined, we should keep expanding on the corpus.

3 Entity sense discovery and embeddings : K-means with variable K

3.1 Data generation

The entity sense discovery data consists of a json file which contains a dictionary mapping between an entity word and a list of sentences where the word is used, an entity word can be used in sentences with different senses. This is the word sense document.

The entity sense embeddings data contains of a pickle file which consists of a dictionary mapping between an entity word and dictonary which contains an numpy array as embedding for every distinct entity sense discovered for the word.

3.2 Methodology

For a word w in the word sense document, sentence embeddings are collected for the corresponding sentences. The last hidden state of the [CLS] token in bert-base-uncased in known to serve as the sentence embedding. Given some K, sentence embeddings are clustered running K-means and the corresponding partition is obtained.

For each partition, for the sentence group associated with the partition, word embeddings of w are obtained from the forword pass of bert, indexing from the final layer hidden states. An aggregate of these word embeddings serves as an entity sense embedding for word w, where the entity sense is represented by the corresponding partition.

Critically, for word w, we do not assume the knowledge of k - number of distinct entity senses. We use silhuette analysis. For each point p in a generated cluster, the average distance a to other points in the same cluster is compared with the minimum average distance b of the point to points in a different cluster. A high value (b-a) can indicate that the data point p is well matched to its own cluster and poorly matched to neighboring clusters. Averaging the values of the silhuette co-efficient across data points can determine the overall effectiveness of the clustering confirguration for some k, enabling iterative comparison with other k values. Torch, sklearn libraries can enable optimized computations. We take into account normalization and regulization.

While silhuette analysis can work well for text data, it requires 2 clusters to begin with. For our purposes, there will be many entities with a single entity sense and thus one cluster. We observe that, if $k >= 2$ is not suitable for an entity, it will have signifcantly low silhuette scores across k values. If $k >= 2$ clustering is suitable for an entity, it will have higher silhuette score, especially for the optimal cluster. After empirical analysis, in consideration of within-cluster-sum-of-squares and silhuette co-efficients, we develop a heuristic with a threshold selected from empirical validation, in order to determine if 1 cluster is suitable for an entity.

3.3 Analysis and Results

For 'amazon',[2] the optimal number of clusters is obtained as 2, the partitioned sentences contain amazon used as a company and amazon used as a forest respectively.

For 'jaguar',[3] the optimal number of clusters is obtained as 2, the partitioned sentences contain jaguar as used a car and jaguar used as an animal respectively.

Similarly, number of distinct entity senses, and corresponding embeddings, seen to align with our semantic notions for apple, palm, army, India, China, star, fifteen etc.

The clustering algorithm recognizes patterns as contingent on the data it is provided and corresponding contexts. This can lead to edge cases. For example, for the word 'mouse', one may expect utility in 2 distinct entity senses, mouse as an animal and the computer mouse. Our algorithm finds 3 distinct entity senses. Upon inspection, it is seen to partition the following; the computer mouse, mouse used in the context of a house, urban settings; wild mouse, sentences present for a wild mouse in forest settings. The U.S. army has a high frequency in the dataset. For the word 'army', it seems to delineate between the U.S. army, other armies. For the word 'orange', it discovers multiple senses for orange as a fruit, orange as a company, and orange as two distinct locations (orange california, orange south africa) as mentioned in the 'orange' word sense document with relative frequency. Considerations on data augmentation and scaling are of utility. One approach is that we cluster on the contextualized word embeddings themselves and maintain a linking of these embeddings with the sentences in order to organize partitions, as apart from clustering on the sentence embeddings. Another approach is that we develop entity type embeddings based the kalmified corpus and make use of these to structure entity sense clusters.

4 Further approaches for Entity sense discovery

The following clustering approaches can be modified for our purposes, have no prior assumptions on the number of clusters, have not been used in previous works for entity sense discovery or embedding generation akin to our formulation.

4.1 Dbscan

Density-based spatial clustering of applications with noise identifies distinct clusters in data without any need for pre-specifying the number of clusters, can potentially be used for our data. Dbscan groups together points that are closely packed, while marking the points that lie alone in low-density regions as outliers.

The main concepts in Dbscan are: 1.Core Points: A point is a core point if a specified number of other points (defined by a threshold minPts) fall within a certain ϵ radius of it. 2.Border Points: Points that are not core points but are within the ϵ radius of a core point.3. Noise Points: Points that are neither core points nor border points. ϵ and minPts are hyperparameters which we can carefully tune for our dataset.

4.2 Graph based methods

One can take inspiration from how node embeddings are developed in methods like deepwalk, node2vec. Words and their contextual features can represent nodes in a graph, with edges in the graph representing co-occurrence frequencies (or semantic relationships, or we could have both).

Graph clustering algorithms to for community detection can be used. Grouping with respect to a vertex, each cluster can be hypothesized to indicate a distinct entity sense of a corresponding word. This is a potentially useful line of inquiry.

5 Further Work and Outline

5.1 Equation-of-thinking

As part of thinking 1.0, in view of entity sense disambiguation, we have to infer a unique combination of given word senses in a sentence.

Given a sentence S as $[w_1[s_1, .., s_{k_1}], w_2[s_1, .., s_{k_2}].....w_n[s_1, ., s_{k_n}]]$.

$P_t(w_i)$ is a distribution over the senses for word i at time step t. For word w_i, the context word set is given by $w_c[i]$.

Let X be a random vector $[X_1, ..., X_n]$ such that X_i is a random variable denoting a distribution over entity senses s_i for word w_i. We will have a transition matrix A which will denote the transition probabilities for iterative updates of X in view of bayes theorem.

Formulations where updates to the beliefs over a variable v are based beliefs over context variables u and influence of u on v, are defined in literature .e.g. pagerank. In view of Markov Chains, the associated matrix computations can be well defined. e.g. under certain conditions on A, X can converge to the dominant eigenvector of A.

For word w_i, context word set $w_c[i]$. The update at an iteration t is given by,

$$P_{t+1}(w_i = s) = \sum_{j \epsilon w_c[i]} (1 - H(P_t(w_c(j)))). \sum_{s' \epsilon w_j} P(w_i = s/w_j = s')P_t(w_j = s')$$

$P(w_i = s/w_j = s')$ is modeled as time step independent. The $(1-H(P_t(w_c(j))))$ term acts a regularizer, indicating the degree of certainty in the current distribution over senses of w_j and subsequent information coming from word w_j. The entropy term can be scaled by $\log(len(w_j[s])) + 1$, the same will ensure the regularizer term lying between 0 and 1. Since $\log(len(w_j[s]))$, corresponding to the uniform distribution over the senses, acts as an upper bound on the entropy $H(P_t(w_c(j))$.

1. How do we define transition probabilities?

A natural starting point would be co-occurence frequencies. The same approach is used in NLP tasks like assigning POS tags with hidden markov models. We will have a large sparse matrix here and will require significant text data management to ensure co-occurence frequency efficiency.

learnable parameters? how to structure the function?

We can define $P(w_i = s/w_j = s')$ in terms of learnable parameters. A neural network architecture can model $P(w_i = s/w_j = s') = f_\theta(h(w_i[s]), h([w_j[s']))$. There can be one encoder which operates on individual entity senses. The similarity score between two sense embeddings can give the transition probability value. There can be another encoder which can operate on the set of entity senses from w_i, w_j as and produce a context vector which can be incorporated in similarity score computations $P(w_i = s/w_j = s') = f_\theta(h(w_i[s]), h([w_j[s']), c)$. We could also use a bilinear form for the similarity score. Efficacy in gradient computations is to be accounted for.

2. What is a suitable modeling framework for inference and learning?

Belief propagation with message passing can be used to perform iterative updates. We can use the concepts in markov networks, cluster graphs and clique trees. One possible clique tree/chain based formulation can be,

For inference, one set of messages will be propagated in from w_1 to w_n. Another set of messages will be propagated in from w_n to w_1. After this is done, the node for w_i will aggregate messages from w_{i-1}, w_{i+1} in accordance with our update step.

For learning, one option is supervised learning, with transition probabilities as parameters, once the relevant distribution is set up, the loss function is set up, one can compute the gradients wrt the transition parameters and subsequently the gradients wrt the parameters of the function which govern the transition parameters.

Apart from having a categorical cross entropy loss, we could think in terms of configurations of states for our loss. For efficient learning, our loss should be able to convey information regarding the closeness of the inference configuration to the label configuration. Our categorical cross entropy loss does the same in a statistical sense. For a given configuration, if we are able to provide a sense making score for that configuration, based on predefined knowledge graphs, we could go for semi supervised learning.

Inference and learning will be performed iteratively.

Another option is Reinforcement learning, value function updates defined iteratively given other value functions have a similar form. Our configuration of entity senses can be a state. Taking an action will entail going from one state to another. Affinity score will be defined across entity sets, looking at the current configuration of states and affinity scores, the policy function can govern transitions in beliefs over entity states for each word.

3. How do we guarantee convergence?

In the clique tree as chain formulation, convergence in message passing for inference is obtained in one forward and backward pass.

We can have other formulations for inference across time steps and learning mechanisms. For a cluster graph, convergence is not necessarily guaranteed.

Reinforcement learning algorithms have convergence criterion defined.

4. Considerations on the context window size, initialization and out-of-vocabulary handling should be elaborated.

5.2 Knowledge module, language modeling and Dekalmification

In the knowledge module an entity and it's relationships are present with disambiguation an unique ids. Ganga as an instance of (P31 wikidata) a river is has a separate id from Ganga as an instance of a person. Each distinct Ganga as person has a distinct id. The difference between instance of and subclass of (P279 wikidata) is adjusted for. Apple(the fruit) is denoted as a subclass of fruit. Apple(the company) is denoted as an instance of organization. The knowledge module and it's ontology is refined and grown independently.

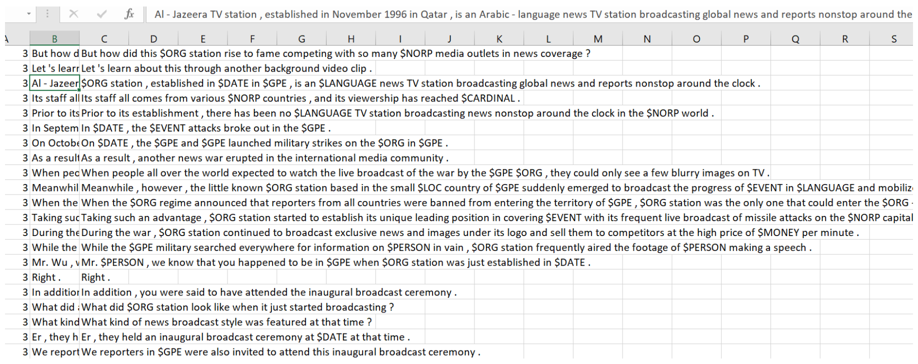
Consider a language model trained on the kalmified corpus and then used(finetuned?) for question answering. Consider the question. "I am studying geography. Where is Ganga located?". Let say us our disambiguation, kalmification modules appropriately kalmify this as "I am studying geography. Where is $river located?"

In order to elaborately generate the output for, "where is $river located?", the information that $river is Ganga is to be effectively incorporated in our inference.

Our language model, predicting a word given the context, should be knowledge agnostic, should be trained on the kalmified corpus, with late knowledge fusion and dekalmification for text generation.

Using this language model, as we build a conversational model, a question answering model, it should incorporate early knowledge fusion with our kalmifed sentence representation for inference.

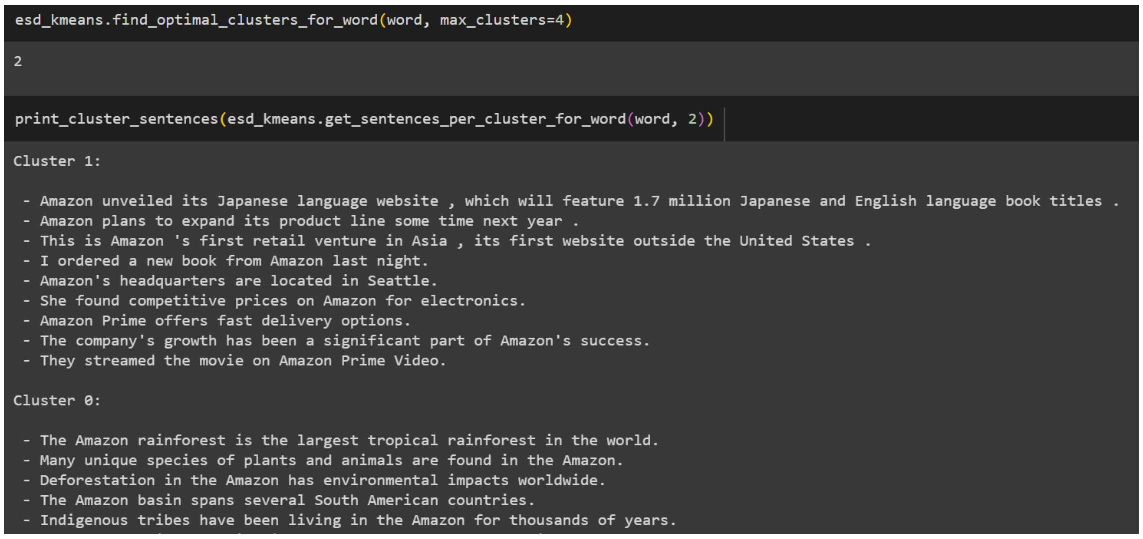
6 Sample figures and code



fx Al - Jazeera TV station , established in November 1996 in Qatar , is an Arabic - language news TV station broadcasting global news and reports nonstop around the

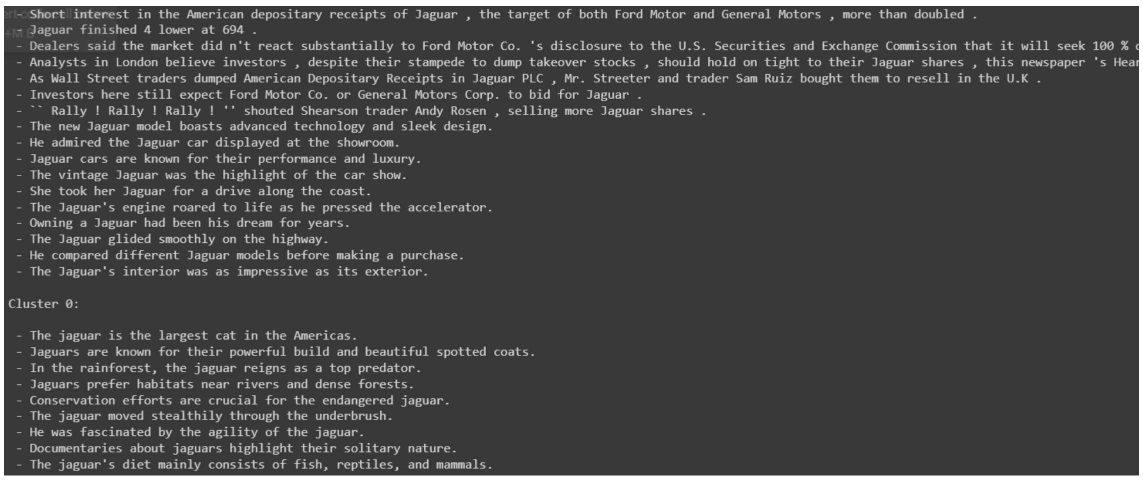
A	B	C
3	But how d	But how did this $ORG station rise to fame competing with so many $NORP media outlets in news coverage ?
3	Let 's learr	Let 's learn about this through another background video clip .
3	Al - Jazeer	$ORG station , established in $DATE in $GPE , is an $LANGUAGE news TV station broadcasting global news and reports nonstop around the clock .
3	Its staff all	Its staff all comes from various $NORP countries , and its viewership has reached $CARDINAL .
3	Prior to its	Prior to its establishment , there has been no $LANGUAGE TV station broadcasting news nonstop around the clock in the $NORP world .
3	In Septem	In $DATE , the $EVENT attacks broke out in the $GPE .
3	On Octob	On $DATE , the $GPE and $GPE launched military strikes on the $ORG in $GPE .
3	As a result	As a result , another news war erupted in the international media community .
3	When peo	When people all over the world expected to watch the live broadcast of the war by the $GPE $ORG , they could only see a few blurry images on TV .
3	Meanwhil	Meanwhile , however , the little known $ORG station based in the small $LOC country of $GPE suddenly emerged to broadcast the progress of $EVENT in $LANGUAGE and mobiliz
3	When the	When the $ORG regime announced that reporters from all countries were banned from entering the territory of $GPE , $ORG station was the only one that could enter the $ORG
3	Taking suc	Taking such an advantage , $ORG station started to establish its unique leading position in covering $EVENT with its frequent live broadcast of missile attacks on the $NORP capital
3	During the	During the war , $ORG station continued to broadcast exclusive news and images under its logo and sell them to competitors at the high price of $MONEY per minute .
3	While the	While the $GPE military searched everywhere for information on $PERSON in vain , $ORG station frequently aired the footage of $PERSON making a speech .
3	Mr. Wu , v	Mr. $PERSON , we know that you happened to be in $GPE when $ORG station was just established in $DATE .
3	Right .	Right .
3	In additior	In addition , you were said to have attended the inaugural broadcast ceremony .
3	What did	What did $ORG station look like when it just started broadcasting ?
3	What kind	What kind of news broadcast style was featured at that time ?
3	Er , they h	Er , they held an inaugural broadcast ceremony at $DATE at that time .
3	We report	We reporters in $GPE were also invited to attend this inaugural broadcast ceremony .

Figure 1: kalm sentences sample



```
esd_kmeans.find_optimal_clusters_for_word(word, max_clusters=4)

2

print_cluster_sentences(esd_kmeans.get_sentences_per_cluster_for_word(word, 2))

Cluster 1:

 - Amazon unveiled its Japanese language website , which will feature 1.7 million Japanese and English language book titles .
 - Amazon plans to expand its product line some time next year .
 - This is Amazon 's first retail venture in Asia , its first website outside the United States .
 - I ordered a new book from Amazon last night.
 - Amazon's headquarters are located in Seattle.
 - She found competitive prices on Amazon for electronics.
 - Amazon Prime offers fast delivery options.
 - The company's growth has been a significant part of Amazon's success.
 - They streamed the movie on Amazon Prime Video.

Cluster 0:

 - The Amazon rainforest is the largest tropical rainforest in the world.
 - Many unique species of plants and animals are found in the Amazon.
 - Deforestation in the Amazon has environmental impacts worldwide.
 - The Amazon basin spans several South American countries.
 - Indigenous tribes have been living in the Amazon for thousands of years.
```

Figure 2: amazon clusters sample



```
- Short interest in the American depositary receipts of Jaguar , the target of both Ford Motor and General Motors , more than doubled .
- Jaguar finished 4 lower at 694 .
- Dealers said the market did n't react substantially to Ford Motor Co. 's disclosure to the U.S. Securities and Exchange Commission that it will seek 100 % o
- Analysts in London believe investors , despite their stampede to dump takeover stocks , should hold on tight to their Jaguar shares , this newspaper 's Hear
- As Wall Street traders dumped American Depositary Receipts in Jaguar PLC , Mr. Streeter and trader Sam Ruiz bought them to resell in the U.K .
- Investors here still expect Ford Motor Co. or General Motors Corp. to bid for Jaguar .
- `` Rally ! Rally ! Rally ! '' shouted Shearson trader Andy Rosen , selling more Jaguar shares .
- The new Jaguar model boasts advanced technology and sleek design.
- He admired the Jaguar car displayed at the showroom.
- Jaguar cars are known for their performance and luxury.
- The vintage Jaguar was the highlight of the car show.
- She took her Jaguar for a drive along the coast.
- The Jaguar's engine roared to life as he pressed the accelerator.
- Owning a Jaguar had been his dream for years.
- The Jaguar glided smoothly on the highway.
- He compared different Jaguar models before making a purchase.
- The Jaguar's interior was as impressive as its exterior.

Cluster 0:

- The jaguar is the largest cat in the Americas.
- Jaguars are known for their powerful build and beautiful spotted coats.
- In the rainforest, the jaguar reigns as a top predator.
- Jaguars prefer habitats near rivers and dense forests.
- Conservation efforts are crucial for the endangered jaguar.
- The jaguar moved stealthily through the underbrush.
- He was fascinated by the agility of the jaguar.
- Documentaries about jaguars highlight their solitary nature.
- The jaguar's diet mainly consists of fish, reptiles, and mammals.
```

Figure 3: jaguar clusters example

Sample code

```
"""
class EntitySenseDiscoveryKMeans core functionality:
find the optimal value of the number of distinct
    entity senses for a word in coprus,
find clusters of sentences for in view of entity sense
     usaage,
find entity sense embeddings for every distinct entity
     sense for a given entity word
"""


   class EntitySenseDiscoveryKMeans:
    def __init__(self, word_sense_document, tokenizer,
         model, device):
         self.word_sense_document = word_sense_document
         self.tokenizer = tokenizer
         self.model = model.to(device) # gpu adjust
             here

    def get_embeddings_for_sentences_of_word(self,
        word):
         """Generate BERT embeddings for sentences
             containing a specific word."""
         sentences = self.word_sense_document.get(word,
            [])
        embeddings = []
        for sentence in sentences:
            inputs = self.tokenizer(sentence,
                return_tensors='pt', truncation=True,
                max_length=512).to(device) # gpu adjust
                 here
            outputs = self.model(**inputs)
            sentence_embedding = outputs.
                last_hidden_state.mean(dim=1)
            embeddings.append(sentence_embedding.
                detach().cpu().numpy()[0]) # gpu adjust
                 here
        return embeddings

    def find_clusters_for_sentences_of_word(self,
       embeddings, num_clusters=2):
        """Cluster sentences using K-means and return
           cluster labels."""
        embeddings_array = np.array(embeddings)
        kmeans = KMeans(n_clusters=num_clusters,
           random_state=0).fit(embeddings_array)
        return kmeans.labels_

    def get_word_embeddings_from_sentences(self, word,
        sentences):
        """Extract contextualized embeddings for a
           specific word from each sentence."""
        word_embeddings = []

        for sentence in sentences:
            # Tokenize the sentence and convert to
                tensor
            inputs = self.tokenizer(sentence,
                return_tensors='pt', truncation=True,
                max_length=512).to(device) # gpu adjust
                 here
            outputs = self.model(**inputs,
                output_hidden_states = True)

            # Identify the token indices corresponding
                 to the word
            tokenized_sentence = self.tokenizer.
                tokenize(sentence)
            word_token_indices = [i for i, token in
                enumerate(tokenized_sentence) if word
            in token]

        # Extract the embeddings for the word
        if word_token_indices:
            # Find the corresponding layers'
                embeddings
            all_layers = outputs.hidden_states
            word_layers = [all_layers[layer][0,
                index] for layer in range(len(
                all_layers)) for index in
                word_token_indices]
            # Average over all layers and word
                occurrences to get a single vector
            word_embedding = torch.mean(torch.
                stack(word_layers), dim=0)
            word_embeddings.append(word_embedding.
                detach().cpu().numpy()) # gpu
                adjust here

    return word_embeddings



def get_sense_embeddings_for_word_from_clusters(
   self, word, cluster_labels):
    """Get average embeddings of the word itself,
        partitioned by cluster."""
    sentences = self.word_sense_document.get(word,
         [])
    word_embeddings = self.
        get_word_embeddings_from_sentences(word,
        sentences)

    unique_labels = set(cluster_labels)
    sense_embeddings = {}
    for label in unique_labels:
        cluster_word_embeds = np.array(
            word_embeddings)[np.array(
            cluster_labels) == label]
        if cluster_word_embeds.size > 0:
            sense_embeddings[label] = np.mean(
                cluster_word_embeds, axis=0)
    return sense_embeddings


def get_sentences_per_cluster_for_word(self, word,
    num_clusters = 2):

   sentences_for_word = self.word_sense_document.
      get(word, [])
   sentence_embeddings = self.
      get_embeddings_for_sentences_of_word(word)
   cluster_labels = self.
      find_clusters_for_sentences_of_word(
      sentence_embeddings, num_clusters)

   sentences_per_cluster = {}
   for label, sentence in zip(cluster_labels,
      sentences_for_word):
        if label not in sentences_per_cluster:
            sentences_per_cluster[label] = []
        sentences_per_cluster[label].append(
           sentence)
   return sentences_per_cluster


def discover_sense_embeddings_for_given_clusters(
   self, word, num_clusters=2):
    """Generate sense embeddings for a given word.
        """
    sentence_embeddings = self.
        get_embeddings_for_sentences_of_word(word)
    cluster_labels = self.
        find_clusters_for_sentences_of_word(
        sentence_embeddings, num_clusters)
    entity_sense_embeddings = self.
        get_sense_embeddings_for_word_from_clusters
        (word, cluster_labels)
    return entity_sense_embeddings

def calculate_wcss(self, word, num_clusters):
    """Calculate the total within-cluster sum of
        square (WCSS)."""
    sentence_embeddings = self.
        get_embeddings_for_sentences_of_word(word)
    kmeans = KMeans(n_clusters=num_clusters,
        random_state=0).fit(sentence_embeddings)
    wcss = kmeans.inertia_
    return wcss

def find_silhouette_score(self,word, num_clusters)
   :
```

```
  sentence_embeddings = self.
     get_embeddings_for_sentences_of_word(word)
  embeddings_array = np.array(sentence_embeddings)
  kmeans = KMeans(n_clusters=num_clusters,
     random_state=0).fit(embeddings_array)
  labels = kmeans.labels_
  silhouette_avg = silhouette_score(
     embeddings_array, labels)

  return silhouette_avg


def find_optimal_clusters_for_word(self, word,
   max_clusters=5):

    """silhouette analysis."""
    sentence_embeddings = self.
       get_embeddings_for_sentences_of_word(word)
    embeddings_array = np.array(
       sentence_embeddings)

    best_score = -1
    optimal_clusters = 2

    for n_clusters in range(2, max_clusters):
        kmeans = KMeans(n_clusters=n_clusters,
           random_state=0).fit(embeddings_array)
        labels = kmeans.labels_
        silhouette_avg = silhouette_score(
           embeddings_array, labels)

        if silhouette_avg > best_score:
            best_score = silhouette_avg
            optimal_clusters = n_clusters

    return optimal_clusters

def
   find_optimal_clusters_sense_embeddings_for_word
   (self, word, max_clusters=5):

    """ find entity sense embeddings for optimal
       number of clusters """

    optimal_entity_sense_embeddings = {}
```

```
sentence_embeddings = self.
   get_embeddings_for_sentences_of_word(word)
embeddings_array = np.array(
   sentence_embeddings)

kmeans_1 = KMeans(n_clusters= 1, random_state
   =0).fit(embeddings_array)
labels_1 = kmeans_1.labels_
entity_sense_embeddings_1 = self.
   get_sense_embeddings_for_word_from_clusters
   (word, labels_1)

best_score = -1
optimal_clusters = 2

for n_clusters in range(2, max_clusters):
    kmeans = KMeans(n_clusters=n_clusters,
       random_state=0).fit(embeddings_array)
    labels = kmeans.labels_
    silhouette_avg = silhouette_score(
       embeddings_array, labels)
    entity_sense_embeddings = self.
       get_sense_embeddings_for_word_from_clusters
       (word, labels)

    if silhouette_avg > best_score:
        best_score = silhouette_avg
        optimal_clusters = n_clusters
        optimal_entity_sense_embeddings =
           entity_sense_embeddings

if(best_score < 0.105):
  # see empirical threshold estimation
  optimal_entity_sense_embeddings =
     entity_sense_embeddings_1


return optimal_entity_sense_embeddings
```