Generating Medical Word Embeddings

Natural Language Processing

July 17, 2020

Introduction

We will give a picture about training and obtaining medical word embeddings. We will discuss about the different medical data that is available. We will have a comparison of the embeddings trained from different methods like CBOW, skipgram etc. and try to give some explanations why each of them might perform in such a way.

Problem Statement

The current problem at hand is to develop an embedding scheme for medical words satisfying the following properties .

- Embeddings for a disease must be close to its corresponding symptoms, medications(drugs), part of the human anatomy etc.
- Given an embedding we must be able to categorize it into few categories human anatomy, disease, symptom , etc. However we have fixed the classes here into disease,symptom and cure.

Apporach - Constraints

There are some constraints that restricted us exploring different options.

- The data that we have procured is unlabelled data, thus restricting us from using supervised methods for classification.
- The current unlabelled data hinders the presence for both the properties for the same embeddings.

Apporach - Options

- One of the main options would be to have a lebelled data where each word to mapped to a class. We can develop a classifier from the embeddings.
- Due to the lack of labelled data we had to breakdown this problem into subproblems. First to create embeddings to satisfy property 1 and then create models/classifiers to satify property 2.

Medical Datasets

Owing to the advancement in the medical field it has become less difficult to obtain medical records , medical texts for developing machine learning models. However most of them do not seem to be available for free. We list some of the well known medical data in textual form.

- **PubMed Dataset**
- MIMIC Dataset
- EBM NLP
- Scraping data from online forums

PubMed Dataset

This data comprises more than 30 million citations for biomedical literature from MEDLINE, life science journals, and online books and RCTs. It has unlabelled corpus with around 200K documents. However, the dataset contains many punctuations and numbers. However one of the versions of the dataset has all the numbers replaces with @ symbol.

Textual Data Preprocessing

One of the main task for NLP is to preprocess data that is obtained from the various sources.In current scenario we have 2 sets of PubMed data 20K, 200K based on the documents covered. Some of the typical preprocessing techniques we used are as below

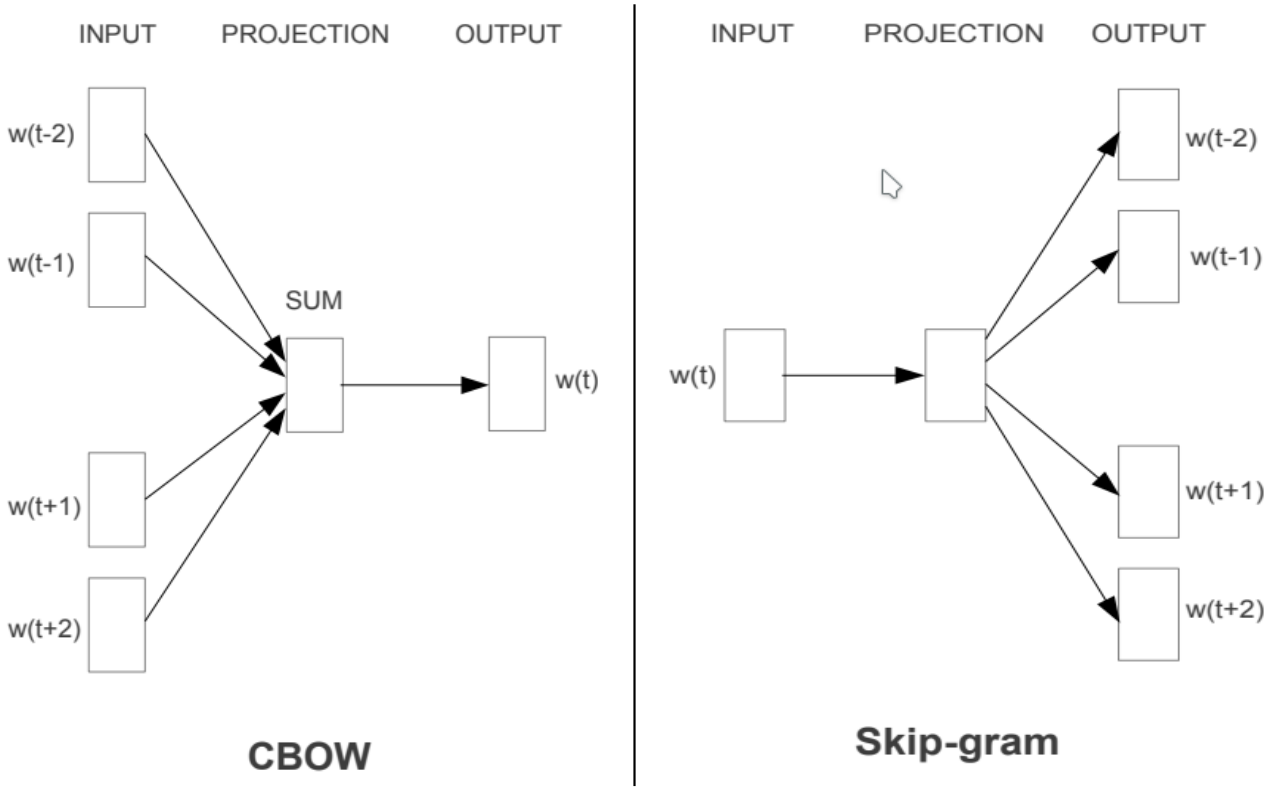
- Removing Stop words and numerical data.
- Lowercasing the words
- Lemmetization and Stemming were a possible task to be done, but there might be change in context A treats B $\rightarrow$ B is treated by A. Therefore we have not gone with them.

Word2Vec

Word2Vec is an architecture proposed by Google research in 2013 for computing vector representations of word on very large datasets. In their paper, they putforth two model architectures namely CBOW and Skipgram for the task with quite opposite learning methods.

- CBOW: Tries to predict the current word based on the context (set window of words before and after the current word).
- SkipGram: Tries to predict the surrounding words (within a set window) given the current words.

CBOW & Skipgram architectures



Source: Efficient Estimation of Word Representations in Vector Space

CBOW & Skipgram - Experiments

For our experiments, we have used the Word2Vec model from the **Gensim** library which has the tools for training and some inference techniques for both CBOW and Skipgram models. We trained the CBOW and Skipgram models using the PubMed dataset.
Following are the experiments we have done:

- Window size - 3,5, and 7.
- With and without removal of stop words.
- With and without lower casing all words.

CBOW - Examples

Some examples of CBOW word embeddings that are closest to word embeddings of:

```
[('CVD', 0.8057840466499329),
 ('atherothrombotic', 0.7619547247886658),
 ('cerebrovascular', 0.7502428293228149),
 ('macrovascular', 0.7423928380012512),
 ('cardiometabolic', 0.7204415202140808),
 ('CHD', 0.7027085423469543),
 ('vascular', 0.6984412670135498),
 ('cardiac', 0.6910381317138672),
 ('thrombotic', 0.6840763092041016),
 ('thromboembolic', 0.6602818965911865),
 ('Cardiovascular', 0.6329425573348999),
 ('cardio-metabolic', 0.6171876192092896),
 ('coronary', 0.6088035106658936),
 ('thrombo-embolic', 0.5994089841842651),
 ('arrhythmic', 0.5947771668434143)]
```

(a) Cardiovascular

```
[('cutaneous', 0.7127124071121216),
 ('Skin', 0.7032967209815979),
 ('facial', 0.6588599681854248),
 ('dermal', 0.6499264240264893),
 ('thermal', 0.635189414024353),
 ('eyelid', 0.6264931559562683),
 ('dentinal', 0.6177009344100952),
 ('erythema', 0.6150423288345337),
 ('wound', 0.6118344664573669),
 ('dentine', 0.60770583152771),
 ('pigmentation', 0.6013128757476807),
 ('conjunctival', 0.6009274125099182),
 ('tongue', 0.6000021696090698),
 ('conjunctiva', 0.5998902320861816),
 ('tooth', 0.5936832427978516)]
```

(b) Skin

Skipgram - Examples

Some examples of Skipgram word embeddings that are closest to word embeddings of:

```
[('CVD', 0.8593591451644897),
 ('CHD', 0.7907732129096985),
 ('macrovascular', 0.780956506729126),
 ('cerebrovascular', 0.779969334602356),
 ('cardio-metabolic', 0.7702720165252686),
 ('CVEs', 0.7678258419036865),
 ('noncardiovascular', 0.7629328966140747),
 ('CVDs', 0.7549433708190918),
 ('atherothrombotic', 0.7514378428459167),
 ('nonatherosclerotic', 0.7478843331336975),
 ('cardiometabolic', 0.7432365417480469),
 ('CVRFs', 0.7429026365280151),
 ('non-cardiovascular', 0.7402695417404175),
 ('CVD-related', 0.7304717302322388),
 ('lifestyle-related', 0.7303916811943054)]
```

(c) Cardiovascular

```
[('cutaneous', 0.8287128210067749),
 ('Skin', 0.7847676277160645),
 ('sunburn', 0.7182192802429199),
 ('dermal', 0.7150313258171082),
 ('eczematous', 0.6894139051437378),
 ('pruritic', 0.6872032284736633),
 ('fingertip', 0.684495747089386),
 ('photoageing', 0.6837857961654663),
 ('immediate-type', 0.6751985549926758),
 ('hyperkeratotic', 0.6669816970825195),
 ('sun-damaged', 0.6669158935546875),
 ('non-melanoma', 0.6645284295082092),
 ('Mantoux', 0.6639062762260437),
 ('periwound', 0.663240909576416),
 ('corneum', 0.6620181798934937)]
```

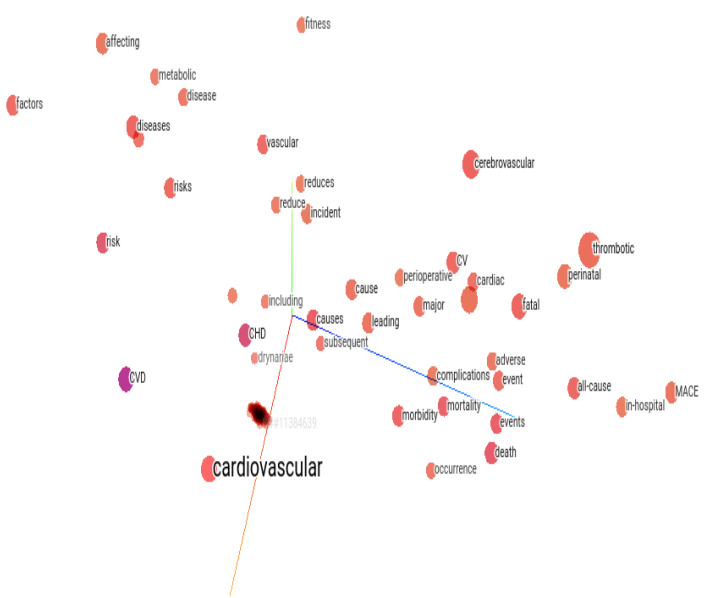
(d) Skin

Glove

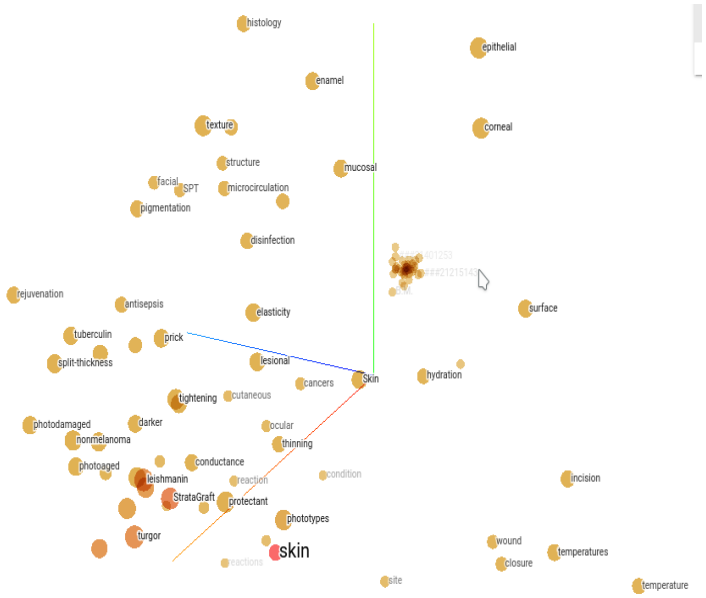
- Glove is another word vector model developed by Stanford NLP. It uses the co-occurrence matrix of words within the selected context and generates word embeddings using this statistical information.
- Instead of using the co-occurrence probabilities of words directly, the paper proposes to use the ratio of co-occurrence probabilities of words since the later case helps in distinguishing between relevant and irrelevant words better.
- The main objective function for Glove is designed by adding some constraints and corrections to the above criterion.

Glove - Examples

We trained Glove model using the glove library in python. We were able to train 100 dimensional vectors for PubMed dataset with context window as 5, 10, and 15. Some examples of Glove word embeddings in tensorflow projector and their closest words:



(e) Cardiovascular



(f) Skin

FastText

Fastext is an extension of the Word2Vec model. It was developed by Facebook AI and research (FAIR).
The basic idea of FastText is to use generate the embeddings for chartacter n grams rather than embeddings for words. Once we get these embeddings the final embedding of the whole word is given by the summation of the embeddings of all n grams of the word.[1]

$$\text{apples} = < ap, app, ppl, ple, les, es >$$

3-grams for the word apples, $<, >$ are the ending tokens for a word.

FastText

- Once a word is split and represented using the character *n-grams*, a skipgram model is trained to learn the embeddigns.

```
model.get_nearest_neighbors('heart',20)
```

```
[(0.7720064520835876, 'heart-failure'),
 (0.7541844844818115, 'congestive'),
 (0.739101231098175, 'Heart'),
 (0.7043959498405457, 'arterial-cardiac'),
 (0.6951054930686951, 'heart/lung'),
 (0.6903347373008728, 'asystolic'),
 (0.6862385869026184, 'systolic'),
 (0.6793137788772583, 'CHF'),
 (0.6762699484825134, '@bpm'),
 (0.6746391654014587, 'cardiac'),
 (0.6738244891166687, 'non-heart'),
 (0.6718320250511169, 'beating-heart'),
 (0.6714013814926147, 'rate-systolic'),
 (0.6692805290222168, 'SIHD'),
 (0.6637172102928162, 'failure'),
 (0.6624863147735596, 'left-ventricular'),
 (0.6611158847808838, 'cardio-respiratory'),
 (0.6608555912971497, 'rate-pressure'),
 (0.653653085231781, 'HF'),
 (0.653373122215271, 'cardiocirculatory')]
```

(g) Heart

```
model.get_nearest_neighbors('nose',20)
```

```
[(0.8252351880073547, 'nose/nasal'),
 (0.806947648525238, 'runny'),
 (0.7981841564178467, 'sneezing'),
 (0.7970660924911499, 'stuffiness'),
 (0.7968831658363342, 'nose/throat'),
 (0.7911538481712341, 'stuffy'),
 (0.782988429069519, 'rhinorrhea'),
 (0.7792298197746277, 'nasal'),
 (0.7725257277488708, 'rhinorrea'),
 (0.7685657143592834, 'bunny'),
 (0.7679197788238525, 'nonnasal'),
 (0.7637228965759277, 'nostrils'),
 (0.7570949196815491, 'noses'),
 (0.7515284419059753, 'postnasal'),
 (0.7488957643508911, 'blows'),
 (0.7484537959098816, 'rhinorrhoea'),
 (0.7475621700286865, 'sneeze'),
 (0.7384189963340759, 'nostril'),
 (0.7373910546302795, 'itchy'),
 (0.731083333492279, 'sneezes')]
```

(h) Nose

FastText

- This model has a the size of the ngram as an additional hyperparameter.

```
model.get_nearest_neighbors('throat',20)
```

```
(0.9268910884857178, 'sore'),
(0.921187162399292, 'nose/throat'),
(0.9205098748207092, 'throats'),
(0.8450037240982056, 'hoarseness'),
(0.8071814775466919, 'Hoarseness'),
(0.7631379961967468, 'pharyngolaryngeal'),
(0.7446455359458923, 'Throat'),
(0.7406995296478271, 'pharyngotonsillitis'),
(0.7251002788543701, 'laryngo-pharyngeal'),
(0.7204555869102478, 'stuffiness'),
(0.7202932834625244, 'expectorating'),
(0.715045690536499, 'cough'),
(0.7102416753768921, 'nose'),
(0.710084080696106, 'tonsillopharyngitis'),
(0.7055957317352295, 'earache'),
(0.700631320476532, 'ear-nose-throat'),
(0.7000788450241089, 'rhinorrhea'),
(0.6964247226715088, 'expectoration'),
(0.6922354102134705, 'sneezing'),
(0.6902430653572083, 'laryngopharyngeal')]
```

(i) Throat

FastText

- One of the advantage of FastText over traditional models is that it can handle OOV words as well. For example,

  If we have a new word *neurology* intuitively, it will be given an embeddigns based on the embedding for "*neuro*" (brain) and "*ology*" which (means study of).

Enhancing embeddings using MeSH

- Recently, some studies(12,13,14,15) have suggested that integrating domain knowledge with the text corpora can be beneficial to improve the quality of word embeddings(more emphasis on sub-word embeddings).
- We construct a MeSH(medical subject headings) term graph based on the MeSH RDF data, followed by a random sampling strategy to generate a number of MeSH term sequences
- Sampling can be done using BFS(breadth first search) or DFS(Depth first search). The pre-existing embeddings are trained on these sequences using transfer learning.
- Due to computational constraints we were unable to produce enough sequences to make a considerable difference.

Classification using a Neural Network

- A Neural network for classifying a given word vector into a disease or a symptom or a cure.
- This Neural network consists of three convolution layers, 128 feature maps per convolution layer with filter sizes 3 and 4 and 5, two dense layers and one fully connected layer with relu activation function and a dropout of 0.5 with binary cross entropy loss function.(The same neural network used by Yoon Kim (2014))
- Unfortunately we could not find enough data to train this network and the results obtained were not comparable to others so we are omitting those results in our presentation.

Classifying based on Cosine Similarity

- We consider the cosine similarity of a word with the words like "disease", "symptom" and "cure".
- We compute the cosine similarity of each word with these three words which represent the classes and the label for the word is assigned based on the class word with which it had maximum cosine similarity.
- This method works for the embeddings generated based on the context of the word(FastText,Word2Vec).

Labelling Data using NER

- NER[5] seeks to locate and classify named entities mentioned in unstructured text into pre-defined categories
- We could use a Named Entity Recognition on the model to identify the words belonging to specific class, thus generating a supervised data.
- We can use different NER models for each of the classes to determine the labels of words for each class and then use some classification model.

Classification by clustering the sentence embeddings

- In this sentences are created in the following way. For every word w

  "w is a disease" , "w is a symptom" , "w is a cure"
- Idea is after we obtain sentence embeddings of these created sentences using pre-trained BERT, these sentences can be clustered into 2 different clusters based on their truth value(a true cluster and a false cluster).
- Labelling a cluster into a true cluster or a false cluster could be done by using the sentences formed using labelled words.
- The challenge of a single word having multiple labels can be resolved by the distance from the centroid of the cluster.
- Unfortunately due to computational contraints we were unable to perform the clustering and hence the above mentioned method is not used for categorizing words in this project

Evaluation Metric - Metric for Property 1

- It is based on cosine similarities.
- We consider a fixed set of diseases, their corresponding symptoms and drugs.
- For every word we sum up the similarities to other words in its trio(a disease, it's symptom and it's drug) and average it on the total number of words in the test data.
- The value of the metric will be higher when the cosine similarity is higher among each trio(a disease, it's symptom and it's drug) in the test dataset i.e., every word is closer to other words in its trio.
- Note that this is not an absolute measure for the property 1 of the problem statement but can be used to compare different embeddings.

Evaluation Metric - Metric for Property 2

- This is a basic accuraccy metric to measure the number of correct classifications as compared to the total number of words in the dataset.

Results

Table: Here are the results for the experiments

Embeddings	Classification method	Property 1	Property 2
CBOW	Cosine similarity	0.5747	—
SkipGram	Cosine similarity	0.5810	0.3555
Glove	Cosine similarity	0.5049	—
Fastext	Cosine similarity	0.6274	0.4647
CBOW	NER	0.5747	0.6361
SkipGram	NER	0.5810	0.6732
Glove	NER	0.5049	0.6777
Fastext	NER	0.6274	0.7132

- Note : These values are calculated on a small test data set of 50 data points(each datapoint refers to a disease, it's symptom and it's cure).

Further Improvements

- In case we acquire some labelled data or we come up with some unsupervised classification models, then we can develop a classification model that takes the embedding as an input and gives the classification score.
- Another extension of the above solution is to somehow incorporate the loss function of the label of the embedding into the loss function of the word embedding model if feasible.
- Using random walks to sample sequences from the MeSH graph.
- Current Fasttext model uses word2vec algorithm, but one could always extend it some other algorithms.

References

- Piotr Bojanowski, Edouard Grave Armand Joulin, Tomas Mikolov *"Enriching Word Vectors with Subword Information"*
- Word2Vec Gensim documentation: https://radimrehurek.com/gensim/models/word2vec.html
- Glove source code: https://nlp.stanford.edu/projects/glove/
- Glove source code: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6758924
- NER Paper *A Survey on Deep Learning for Named Entity Recognition*