Advanced NLP (Monsoon 2021)

Word Embedding Models

Rohan Bennur

Word Embedding Model 1

Word Embedding model using Co-Occurrence matrix by applying Singular Value Decomposition (SVD)

In this question, the whole corpus was used to train the model. There were around **16,89,188** review text paras (objects in the corpus' JSON file) in the corpus which themselves had a varying number of sentences. After preprocessing the corpus using regular expressions (**re**) and tokenizers (**nltk.tokenize**), the vocabulary class was created with the following items,

- **word2ind**
- **ind2word**
- **word2freq**

These mappings are necessary because the training of the model requires input to be in the format of numbers because algorithms cannot process raw text. Hence, these mappings are used to convert the input texts/words to numbers which the algorithm can process and hence, they can be used further for training the models.

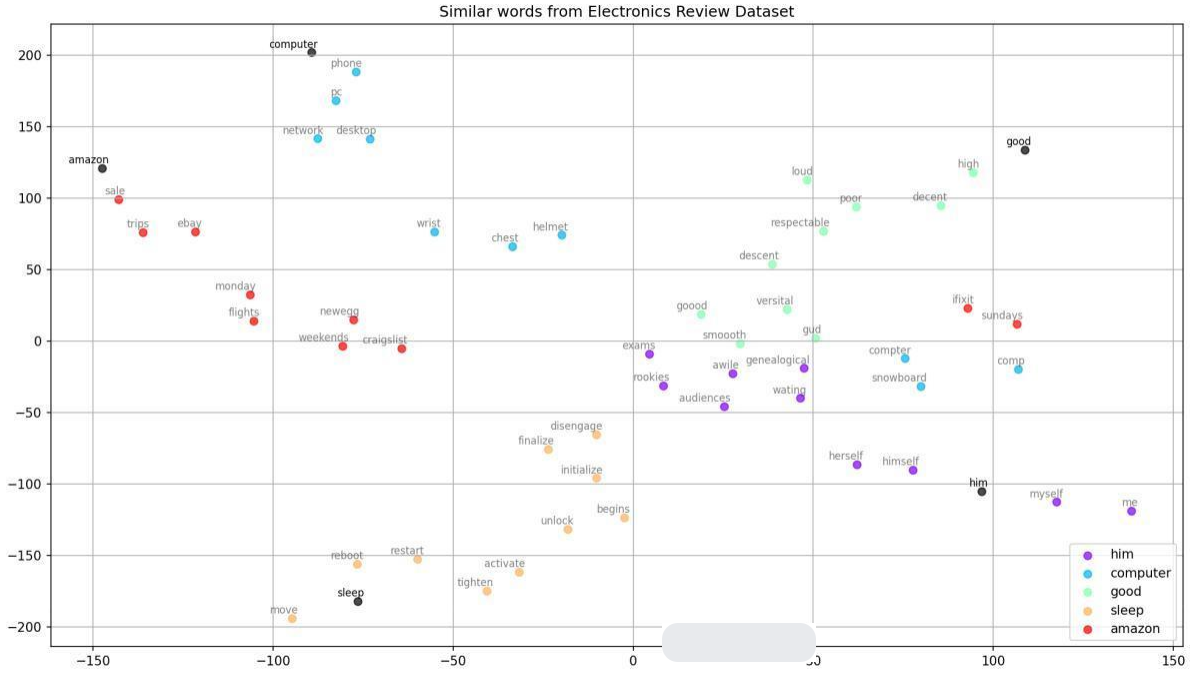
However, as the corpus had many low-frequency words, they were removed in one model. In another model, they were retained in the vocabulary to compare the performance. The vocabulary size was around **5,11,593** without removing low-frequency words and it was around **1,00,718** after removing low-frequency words from the vocabulary and corpus. The threshold on frequency for the token to be considered was kept as **5**.

Model is trained using the **Co-Occurrence** matrix by applying **Singular Value Decomposition (SVD)**. As the vocabulary size is too large, it requires a lot of RAM to store the Co-Occurrence matrix of the size **vocabsize x vocabsize**. However, we know that this matrix is sparse and hence, it can be compressed using libraries. Hence, I used the Scipy library's sparse **csr_matrix** module to encode the values of the Co-Occurrence matrix into a sparse matrix. Scipy provides inbuilt methods to compute the SVD on its sparse matrix object and hence, the embedding vectors were obtained by applying SVD on the sparse Co-Occurrence matrix using Scipy. The SVD produces 3 matrices namely, U, S and Vt in which the U matrix of the shape **vocabsize x embedsize** is the embeddings of all tokens in the vocabulary whose size is **embedsize**.

Various methods are implemented in the Model class to obtain the word embeddings given the input word and the top n similar words given the root word and number n.

The folder contains the following items,

- Trained model instance (**model.pt**)
- Vocabulary (**vocab.pt**)
- Word Embeddings (**embed.npy**)
- t-SNE plot (**plot.png**)



We can see that most of the similar words of the root word are well clustered and organised, except for some. We can see that a few similar words obtained from the model for the words ‘amazon’ and ‘computer’ are a little far away from their respective clusters. This could be due to the PCA algorithm which is representing the word embeddings in reduced dimension space or poorly learnt word embeddings.

The keywords used to evaluate similar words from the model are

```
['him', 'computer', 'good', 'sleep', 'amazon']
```

Similar words for the keyword ‘**him**’ with their cosine similarity scores are

```
[('him', 1.0000000000000002),
 ('himself', 0.8573776513596217),
 ('me', 0.8493789776175894),
```

```
 ('myself', 0.8489974065097141),
 ('awile', 0.8466326788069042),
 ('herself', 0.8244543249256534),
 ('wating', 0.8138127608790332),
 ('audiences', 0.8104827370778579),
 ('exams', 0.8066464002742654),
 ('genealogical', 0.8007336643913884),
 ('rookies', 0.7992310834400729)]
```

Similar words for the keyword '**computer**' are

```
[('computer', 0.9999999999999998),
 ('pc', 0.9809905160993548),
 ('phone', 0.9719092536574497),
 ('comp', 0.9670957169292325),
 ('network', 0.9597486651391246),
 ('helmet', 0.9577643425489787),
 ('wrist', 0.9536278703079802),
 ('chest', 0.9491422096522824),
 ('desktop', 0.9431241057930885),
 ('compter', 0.9409200868331677),
 ('snowboard', 0.9405095009719925)]
```

Similar words for the keyword '**good**' are

```
[('good', 1.0),
 ('high', 0.8403219099979169),
 ('poor', 0.8094442269868936),
 ('gud', 0.769117215385761),
 ('respectable', 0.7559225980254826),
 ('versital', 0.7457933088203441),
 ('descent', 0.7366880997242456),
 ('loud', 0.7353858541722905),
 ('smoooth', 0.7171960635469167),
 ('decent', 0.7051304370269718),
 ('goood', 0.6984927521701176)]
```

Similar words for the keyword '**sleep**' are

```
[('sleep', 1.0),
 ('finalize', 0.8743541215162405),
 ('activate', 0.8250019934167477),
 ('begins', 0.8230901591767842),
 ('unlock', 0.8190145990123429),
 ('tighten', 0.8146018427724797),
 ('restart', 0.8142516917793456),
 ('initialize', 0.8139237117273143),
 ('reboot', 0.7939080451676324),
 ('disengage', 0.787974157983493),
 ('move', 0.7874432682273542)]
```

Similar words for the keyword '**amazon**' are

```
[('amazon', 0.9999999999999999),
 ('ebay', 0.9679187685667973),
 ('flights', 0.9672623165041754),
 ('craigslist', 0.9661323147517309),
 ('newegg', 0.9634297974669861),
 ('sundays', 0.9623034396977448),
 ('weekends', 0.9613231635171653),
 ('ifixit', 0.9601299729048159),
 ('monday', 0.9592043443031376),
 ('trips', 0.9591516048160236),
 ('sale', 0.9570144337720387)]
```

I have obtained the **top 11** words for each keyword because the first word will always be the keyword itself with the highest **cosine similarity ~ 1**. Hence, we get the top 10 distinct similar words other than the keyword itself.

We can see that most of the words obtained from the model for each keyword are almost similar to the keyword in terms of **meaning** representation based on **context**. Let's analyze the top 10 similar words obtained from the model for each keyword.

For the keyword '**him**', we can see that many words like '**himself**', '**me**', '**myself**' and '**herself**' are similar because of the context tag '**pronouns**'. These 4 words are similar in terms of **context** representation because all 4 are different forms of **pronouns** (imperative, possessive etc). The rest of the words most probably appear frequently in the context window of this keyword and hence the **SVD** has recognised them as similar words to the keyword.

For the keyword '**computer**', we can see that many words are similar in terms of meaning representation like '**pc**', '**comp**', '**desktop**', '**compter**'. These 4 words are similar to the keyword in terms of **meaning** representation. Some other words such as '**network**', '**phone**' represent similar **context** representations of the keyword. They are related to the keyword under the contextual tag '**Electronics**'. The rest of the words must have appeared in the context and hence, they occur in the top **10** similar words list for the keyword '**computer**'. But however, an interesting fact is that the cosine similarities for these words are also high which is unlikely.

For the keyword '**good**', we can see that many words are similar to the keyword. Words like '**respectable**', '**gud**', '**descent**', '**smooth**', '**decent**', '**goood**' are similar to the keyword in terms of **meaning** representation. Some words like '**high**', '**poor**' and '**loud**' seem to **antonyms** to the keyword in some **context** representation. Being **high** or **loud** sometimes can be considered bad and hence are antonyms in these contexts. Being poor doesn't imply good and hence, might not be related much to the keyword in terms of **meaning** representation. The rest of the words could have appeared in the context.

For the keyword '**sleep**', we can see that many words such as '**activate**', '**begins**', '**unlock**', '**restart**', '**initialize**', '**reboot**', '**disengage**' are similar to the keyword in terms of **context** representation. These all words are referred to in computer systems related to system **sleep**, **shut down** and **turn on**. And since the corpus is on electronics-related data, the model has learnt meaning representations for the keyword '**sleep**' in the context of the **computer**. The rest of the words could have been learnt based on appearance in the context window and similar context representations.

For the keyword '**amazon**', we can see that some words like '**ebay**', '**weekends**', '**monday**', '**sale**' are similar because of the context '**online shopping**'. Some other words like '**trips**', '**flights**' could have appeared because of context similarity with **online booking** (which includes shopping as well as travelling and trips). The rest of the words could have appeared due to similar contextual learning.

The top 10 closest words to the word '**camera**' obtained from the trained model using the Co-Occurrence matrix by applying SVD are as follows,

```
[('camera', 0.9999999999999998),
 ('cam', 0.9371049060515714),
 ('psu', 0.921371199543117),
 ('projector', 0.9202497342633565),
 ('detector', 0.9163765162074519),
 ('hu', 0.914285371611892),
 ('headset', 0.9119752431055327),
 ('router', 0.8968091957484156),
 ('player', 0.8877756433196792),
 ('radio', 0.8823712782394749),
 ('lens', 0.8796754497812865)]
```

The top 10 closest words to the word '**camera**' obtained from the pre-trained Word2Vec model using gensim (`word2vec-google-news-300`) are as follows,

```
[('cameras', 0.8131939172744751),
 ('Wagging_finger', 0.7311819791793823),
 ('camera_lens', 0.7250816822052002),
 ('camcorder', 0.7037474513053894),
 ('Camera', 0.6848659515380859),
 ('Canon_digital_SLR', 0.6474252939224243),
 ('Cameras', 0.6350969076156616),
 ('Nikon_D####_digital_SLR', 0.6259366273880005),
 ('tripod', 0.6189837455749512),
 ('EyeToy_USB', 0.617348670959472 7)]
```

We can see that the similar words generated by the pretrained **Word2Vec** model (**gensim**) are more similar to the root word 'camera' than those generated by the **Co-Occurrence** matrix model by applying **SVD**.

From our model (**Co-Occurrence** matrix with **SVD**), we can see that some words like '**cam**', '**projector**', '**lens**' are related to the keyword '**camera**' in terms of meaning representation. However, some words like '**headset**', '**player**', '**radio**' belong to '**music**' in terms of contextual meaning, but they have appeared here maybe because of context information which correlated music and camera in the corpus. Some other words like '**detector**', '**router**' are related to network and computer, but they too have appeared here maybe because of contextual correlation between these words. The rest of the words might have appeared due to similar context representation learning.

From pretrained **Word2Vec** word embeddings (**gensim**), we can see that almost all words are related to camera and photography except words like '**Wagging_finger**', '**EyeToy_USB**'. '**EyeToy_USB**' could be something related to the USB cable used in the camera, but the prefix looks a little off-topic. Hence, apart from these **2** words, the rest **8** words are related to camera or photography.

Hence, we can see that the pretrained **Word2Vec** word embeddings extract better **top 10** similar words to the keyword '**camera**' than our model which is trained using **Co-Occurrence** matrix with **SVD**. This is because the pretrained gensim **Word2Vec** model is trained on a large corpus and it uses **neural network architecture** to learn the word embeddings which learns better because of its nature. **SVD** on the other hand is powerful but it learns weak representations of the word meanings when compared to **neural networks**.

Word Embedding Model 2

Word Embedding model using Word2Vec with Continuous Bag of Words (CBOW)

In this question, the partial corpus was used to train the model due to computational limitations. There were around **16,89,188** review text paras (objects in the corpus' JSON file) in the corpus, out of which, only the first **1,00,000** (1 Lakh) review text paras were used to train the model. However, the vocabulary was created using the complete corpus to get better results. After preprocessing the corpus using regular expressions (**re**) and tokenizers(**nltk.tokenize**), the vocabulary class was created with the following items,

- **word2ind**
- **ind2word**
- **word2freq**

These mappings are necessary because the training of the model requires input to be in the format of numbers and algorithms cannot process raw text. Hence, these mappings are used to convert the input texts/words to numbers and hence, they can be used further for training the models.

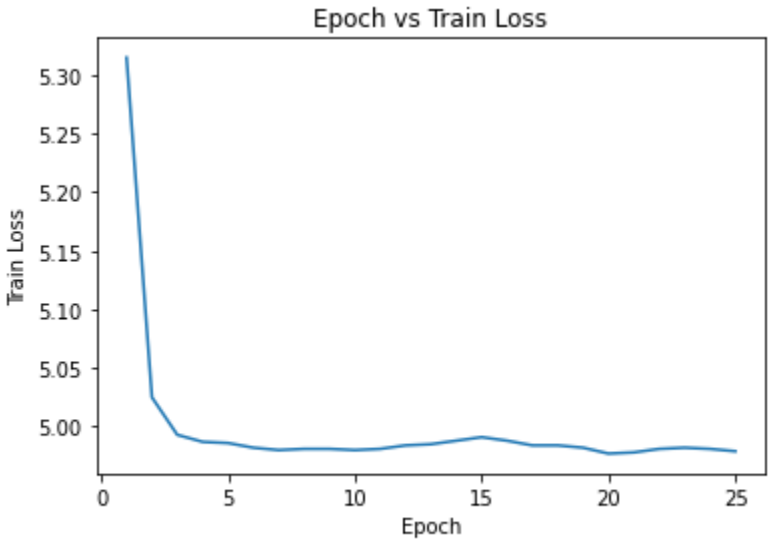
However, as the corpus had many low-frequency words, they were removed from the vocabulary and the corpus. The vocabulary size was around **5,11,593** before removing

low-frequency words and it was around **1,00,718** after removing low-frequency words from the vocabulary and corpus. The threshold on frequency for a token to be considered was kept as **5**.

Model is trained using the **Word2Vec** with **CBOW** method, which uses a continuous bag of words (**context**) information to predict the middle word. A larger context gives more information to the predicted middle word. As this is compute-intensive, only the first 1 Lakh review text paras in the corpus were used. **PyTorch** modules are used to implement the **Word2Vec** model class with methods to train, get the **embeddings** and get the **top n similar words**. The first layer's weight matrix depicts the word embeddings of all the words in the vocabulary, which is of the size vocabsize x embedsize. The input is batched using the PyTorch **Dataset** and **DataLoader** modules.

The **parameters** used while training the model are as follows,

```
{
    batch_size: 64,
    context_size: 2 (window size: 5),
    epochs: 25,
    learning_rate: 0.001,
    optimizer: Adam,
    loss: CrossEntropyLoss
}
```

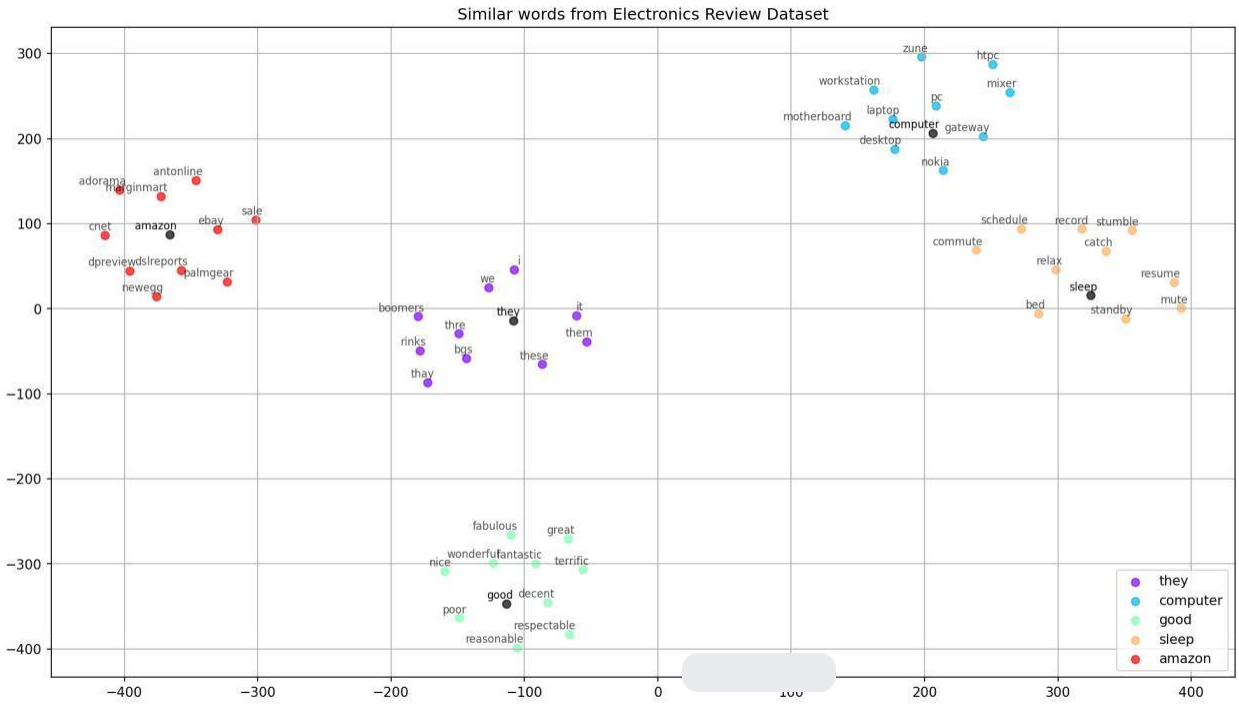


(Epoch vs Train Loss graph: loss.png)

We can see from the **Epoch** vs **Train Loss** graph that the model learns adequate information by epoch **8-10**. Hence, the training loss is not changing drastically after epoch **10**. In fact, it is fluctuating around **4.8 - 4.9** after epoch **10**.

The folder contains the following items,

- Trained model instance (**model.pt**)
- Vocabulary (**vocab.pt**)
- Word Embeddings (**embed.npy**)
- t-SNE plot (**plot.png**)
- Epoch vs Train Loss plot (**loss.png**)
- Logfile obtained from training (**logfile.log**)



We can see that all words are very well clustered to their respective keywords. When comparing this model with the Co-Occurrence with SVD model, we can say that this model does very well because it is evident that well clustering means well-learnt word embeddings. Hence, this shows a better representation of the words as word embeddings in terms of meaning and context. Hence, the **word embedding** vectors learnt from **Word2Vec** with the **CBoW** model has learnt better representation of the words in the reduced dimensional space (word meanings representation is considered to be very high-dimensional).

The keywords used to evaluate word embedding model are

```
['they', 'computer', 'good', 'sleep', 'amazon']
```

Similar words for the keyword '**they**' are

```
[('they', 0.9999999403953552),
 ('we', 0.6522179245948792),
 ('these', 0.5493198037147522),
 ('thay', 0.532634437084198),
```

```
 ('it', 0.5055504441261292),
 ('i', 0.4630432724952698),
 ('boomers', 0.4430449306964874),
 ('bgs', 0.43148767948150635),
 ('rinks', 0.42780786752700806),
 ('them', 0.4231581389904022),
 ('thre', 0.42133012413978577)]
```

Similar words for the keyword '**computer**' are

```
[('computer', 1.0),
 ('pc', 0.8809865117073059),
 ('laptop', 0.859147310256958),
 ('desktop', 0.8499017953872681),
 ('motherboard', 0.7941724061965942),
 ('workstation', 0.7940576076507568),
 ('mixer', 0.7619088292121887),
 ('zune', 0.7519716024398804),
 ('nokia', 0.7431981563568115),
 ('gateway', 0.7386454343795776),
 ('htpc', 0.7317997217178345)]
```

Similar words for the keyword '**good**' are

```
[('good', 1.0),
 ('decent', 0.8823120594024658),
 ('great', 0.8078857064247131),
 ('fantastic', 0.7565606236457825),
 ('poor', 0.7303208112716675),
 ('reasonable', 0.7036711573600769),
 ('nice', 0.6874144077301025),
 ('wonderful', 0.6863434314727783),
 ('respectable', 0.6710003018379211),
 ('fabulous', 0.6707828640937805),
 ('terrific', 0.6630220413208008)]
```

Similar words for the keyword '**sleep**' are

```
[('sleep', 1.0000001192092896),
 ('mute', 0.7251031398773193),
 ('bed', 0.7234523296356201),
 ('relax', 0.708376407623291),
 ('standby', 0.7067058682441711),
 ('commute', 0.6929498314857483),
 ('catch', 0.6693839430809021),
 ('resume', 0.665733814239502),
 ('record', 0.6591612696647644),
 ('stumble', 0.6540684700012207),
 ('schedule', 0.6482977271080017)]
```

Similar words for the keyword '**amazon**' are

```
[('amazon', 1.0),
 ('newegg', 0.7902311086654663),
 ('ebay', 0.7753193378448486),
 ('cnet', 0.7551423907279968),
 ('dpreview', 0.746418297290802),
 ('adorama', 0.7439237833023071),
 ('dslreports', 0.7349218130111694),
 ('sale', 0.7347566485404968),
 ('antonline', 0.7164697647094727),
 ('marginmart', 0.7109187245368958),
 ('palmgear', 0.6859477162361145)]
```

I have obtained the **top 11** words for each keyword because the first word will always be the keyword itself with the highest **cosine similarity ~ 1**. Hence, we get the top 10 distinct similar words other than the keyword itself.

We can see that most of the words obtained from the model for each keyword are almost similar to the keyword in terms of **meaning** representation based on **context**. Let's analyze the top 10 similar words obtained from the model for each keyword.

For the keyword '**they**', we can see that many words like '**we**', '**these**', '**thay**', '**it**', '**i**' and '**them**' are similar because of the context tag '**pronouns**'. These 6 words are similar in terms of **context** representation because all 6 are different forms of **pronouns** (imperative, possessive etc). The rest of the words most probably appear frequently in the context window of this keyword and hence the **Word2Vec** model has recognised them as similar words to the keyword.

For the keyword '**computer**', we can see that many words are similar in terms of meaning representation like '**pc**', '**laptop**', '**desktop**', '**motherboard**', '**gateway**'. These 4 words are similar to the keyword in terms of **meaning** representation. Some other words such as '**workstation**', '**mixer**', '**nokia**' represent similar **context** representations of the keyword. They are related to the keyword under the contextual tag '**Electronics**'. The rest of the words must have appeared in the context and hence, they occur in the top **10** similar words list for the keyword '**computer**'.

For the keyword '**good**', we can see that many words are similar to the keyword. Words like '**great**', '**fantastic**', '**decent**', '**nice**', '**wonderful**', '**respectable**', '**fabulous**' are similar to the keyword in terms of **meaning** representation. Some words like '**poor**', '**reasonable**' and '**terrific**' seem to **antonyms** to the keyword in some **context** representation. Being **reasonable** or **terrific** sometimes can be considered bad and hence are antonyms in these contexts. Being poor doesn't imply good and hence, might not be related directly to the keyword in terms of **meaning** representation. But most probably, the opposite meaning representations are captured by the model. The rest of the words could have appeared in the context.

For the keyword '**sleep**', we can see that many words such as '**bed**', '**relax**' are similar to the keyword in the **contextual meaning** representation related to human sleep. However, there are some other words like '**mute**', '**standby**', '**resume**', '**record**', '**schedule**' that are similar to the keyword in terms of **contextual meaning** representation related to computer sleep. Since the corpus is on electronics-related data, the model has learnt meaning representations for the keyword '**sleep**' in the context of the **computer** also apart from human sleep. The rest of the words could have been learnt based on appearance in the context window and similar context representations.

For the keyword '**amazon**', we can see that some words like '**ebay**', '**cnet**', '**dpreview**', '**dslreports**', '**sale**', '**antonline**', '**marginmart**' are similar because of the context '**online shopping**'. The rest of the words could have appeared due to similar contextual learning.

The top 10 closest words to the word '**camera**' obtained from the trained model using the **Word2Vec** with **CBOW** are as follows,

```
[('camera', 1.0000001192092896),
 ('slr', 0.7598601579666138),
 ('camcorder', 0.7342973351478577),
 ('lens', 0.7333139777183533),
 ('cameras', 0.7170840501785278),
 ('backpack', 0.7101454734802246),
 ('lense', 0.7043905854225159),
 ('film', 0.70195072889328),
 ('dslr', 0.6932461261749268),
 ('bag', 0.6805490851402283)]
```

The top 10 closest words to the word '**camera**' obtained from the pre-trained Word2Vec model using gensim (`word2vec-google-news-300`) are as follows,

```
[('cameras', 0.8131939172744751),
 ('Wagging_finger', 0.7311819791793823),
 ('camera_lens', 0.7250816822052002),
 ('camcorder', 0.7037474513053894),
 ('Camera', 0.6848659515380859),
 ('Canon_digital_SLR', 0.6474252939224243),
 ('Cameras', 0.6350969076156616),
 ('Nikon_D####_digital_SLR', 0.6259366273880005),
 ('tripod', 0.6189837455749512),
 ('EyeToy_USB', 0.6173486709594727)]
```

We can see that the similar words generated by the pretrained **Word2Vec** model (**gensim**) are more similar to the root word 'camera' than those generated by the **Co-Occurrence** matrix model by applying **SVD**. However, our model (**Word2Vec** with **CBoW**) does better than the previous Q1 model (**Co-Occurrence matrix** with **SVD**)

From our model (**Word2Vec** with **CBoW**), we can see that many words like '**slr**', '**camcorder**', '**lens**', '**cameras**', '**lense**', '**film**', '**dslr**' are related to the keyword '**camera**' in terms of meaning representation. However, some words like '**backpack**', '**bag**' also are similar to the keyword contextually because the corpus could have had data where the backpack or bag was mentioned (camera bags). Hence, almost all words are precisely similar to the keyword '**camera**' in terms of **context meaning** representations.

From pretrained **Word2Vec** word embeddings (**gensim**), we can see that almost all words are related to camera and photography except words like '**Wagging_finger**', '**EyeToy_USB**'. '**EyeToy_USB**' could be something related to the USB cable used in the camera, but the prefix looks a little off-topic. Hence, apart from these **2** words, the rest **8** words are related to camera or photography.

Hence, we can see that the pretrained **Word2Vec** word embeddings extract better **top 10** similar words to the keyword '**camera**' than our model which is trained using **Word2Vec** matrix with **SVD**. Even though our model also does very well, the pretrained **Word2Vec** model does better because it outputs even closer meaning words than our **Word2Vec** with the **CBoW** model. This is because even though both are learnt using **neural networks** the pretrained gensim **Word2Vec** model is trained on a large corpus to learn the word embeddings and hence learns better word embedding representations than our model.

Note: For some similar words obtained from the model for a keyword, cosine similarity values are greater than 1 by a very small margin. This is because of floating-point error while converting from tensor to float type. We can see that the value has zeros (0) for atleast 5 decimal places.