Intro to NLP assignment 2 Report

Navaneet Kumar Singh
2020201064

Negative Sampling:

Negative sampling seeks to maximize similarity between words in the same context while limiting similarity between words in other contexts.

Instead of minimizing all terms in the dictionary except for context words, it selects a small number of words (k = 2 to 20) at random and uses them to optimize the goal, depending on the training size.For smaller datasets, we use a higher k.

In word2vec training, we can utilize either skip gram or cbow to predict the context from the word.

Let's look into cbow architecture as an example of negative sampling.

In most cases, we use context as the train data and the word as the label in cbow.This will only generate the positive samples.

We'll create k random samples with terms that don't appear in the positive samples as the background for negative sampling.

Let's look at a few examples to understand it.

For example we have the sentences ‘sun rises in the east and sets in the west ’. And “he rises to the top’ a very small corpus.

For example we have the sentence ‘I am happy because I am learning english ’. And “he is learning to play cricket’ a very small corpus.

If we take the `learning` as the center word and the ‘is’,’am’, ‘to’ and ‘english’ comes as the context in the positive sense and rises may have several such different such context words in the corpus.

For generating negative samples we will pick random words that are not present in the positive context like ‘kill’, ‘adamant’ , ‘generate’

Now the data set becomes

(am,learning), (english,learning),(is,learning), (to,learning) as positive samples

(kill,learning), (adamant,learning), (generate,learning) are negative samples , we can generate k such negative samples for the center word learning.

As previously stated, cbow architecture is
context—> target words

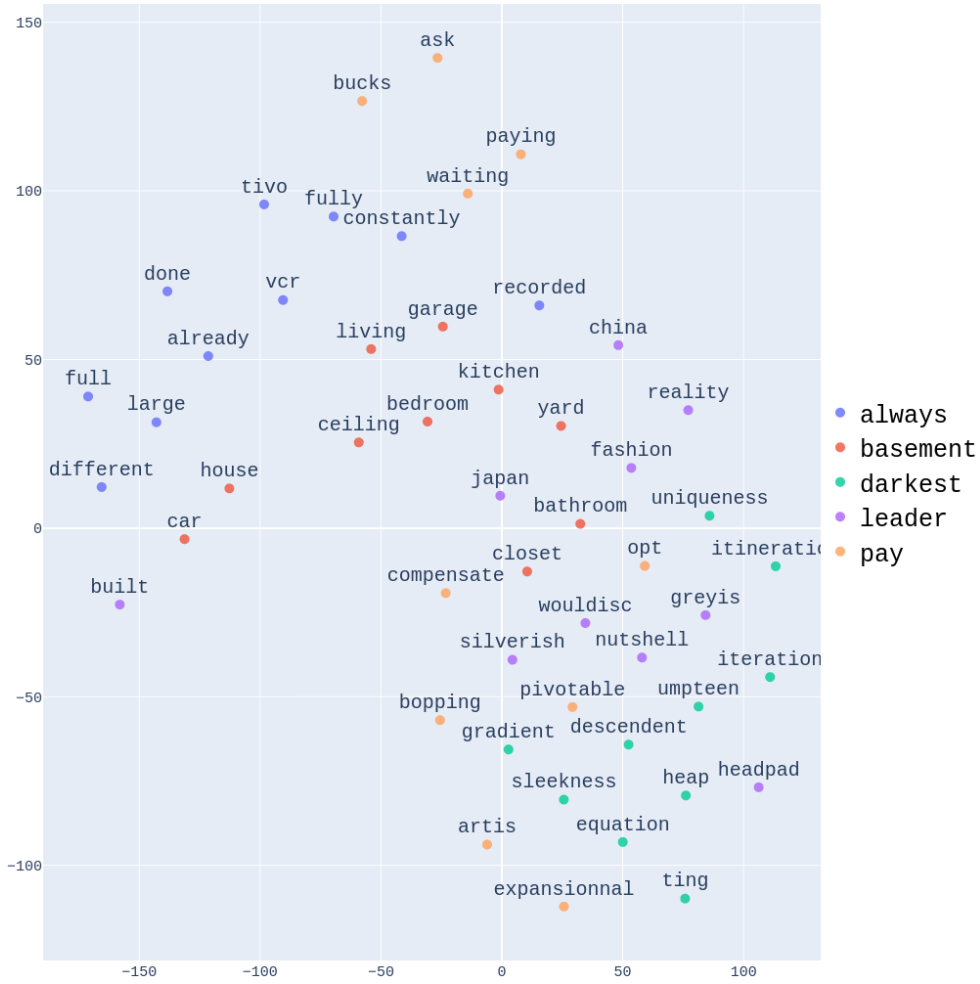
The architecture now switches to binary classification, determining whether the target words are in a positive or negative context.
Positive samples can have a target label of '1' while negative samples can have a target label of '0.'
Another technique can be how much probable that it belongs to positive context and vice versa
(context,target) —-------> probabilities(0.90) or binary classification(1 or 0)

Co occurrence matrix with svd:

To obtain co occurrence matrix we used context in both direction left and rights and the dimensions of the matrix is (29230, 29230)

The reduced matrix, which are our embeddings, was then generated using Truncated svd.
I selected 90 as the dimension of each vector, but any number can be used, and hyper parameter tuning is used to get the best dimension.

Words taken are `sample_words = ['always','basement','darkest','leader','pay']` under different parts of speech . The below figure shows the top to words nearest to the above words



CBOW :

As indicated in the negative sampling description, we generated both positive and negative samples for the Cbow model.

In a second dictionary, we kept the context words for each center word as the set and the center word as the key.

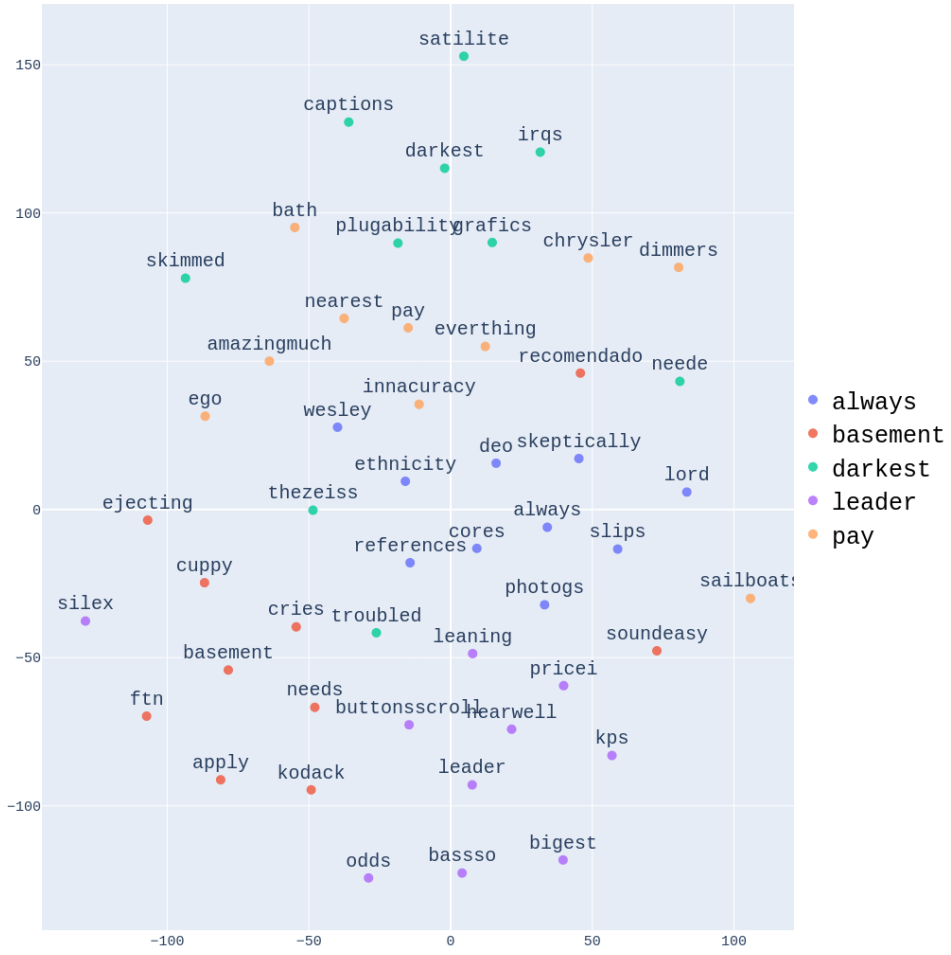
Then, given the available negative indices, we generated 50(k value) negative samples for each word.To acquire the negative indices, we separated the positive indices from the overall indices. If the center word "her" has the context words "she," "women," and so on in positive samples, we can deduct the category encoded is of those two words from all accessible categories to

produce negative ids.We chose four negative ids from that list .To create the negative sample, the target word is attached at the end.

Later we concatenated both positive samples and negative samples to form the Training data set

For labels we took 1 for positive samples and 0 for negative samples.

Words taken `sample_words=['always','basement','darkest','leader','pay']` under different parts of speech . The below figure shows the top to words nearest to the above words



10 nearest word find out by word embedding model

10 nearest word for word 'camera' using word embedding model

```
similar[w] = nearest_words(vocab,word2Ind,Ind2word,'camera',num=10)
similar_word.extend(similar[w])
```

```
antenna 0.994439742472162
lens 0.992465781238871
palm 0.9920087607973263
receiver 0.9919921981410513
player 0.9919531330879344
cradle 0.9917315379052744
reader 0.991482346196552
system 0.9913066438809595
hardware 0.9910211613073194
card 0.9906212489478338
```

10 nearest word find out by gensim pretrained model

```
similarity[w] = similarity_gensim('camera',pre_trained_model,num=10)
similar_word.extend(similarity[w])
```

```
blasting 1.0
blatant 0.8474850654602051
obsessive 0.754730224609375
retrodesign 0.6977447867393494
obstacles 0.6788259148597717
hpcs 0.6621578335762024
contacto 0.6336271166801453
filtro 0.6287022233009338
connects 0.6272484064102173
sand 0.6241426467895508
```

10 nearest word find out by word2vec Cbow model

10 nearest word for word 'camera' using cbow model

```
[161] similarity[w] =  similarity_cbow(word2Ind,'camera',embedding_layer,vocab,num=10)
      similar_word.extend(similarity[w])
```

```
camera 1.0
geography 0.49916425347328186
award 0.49798595905303955
automatcially 0.49518898129463196
carryable 0.49262094497680664
settled 0.47833219170570374
higly 0.47606948018074036
actives 0.46999391913414
roundish 0.46856632828712463
adquate 0.464630663394928
```