Intro To NLP

Assignment 2 Report

Question 2.1

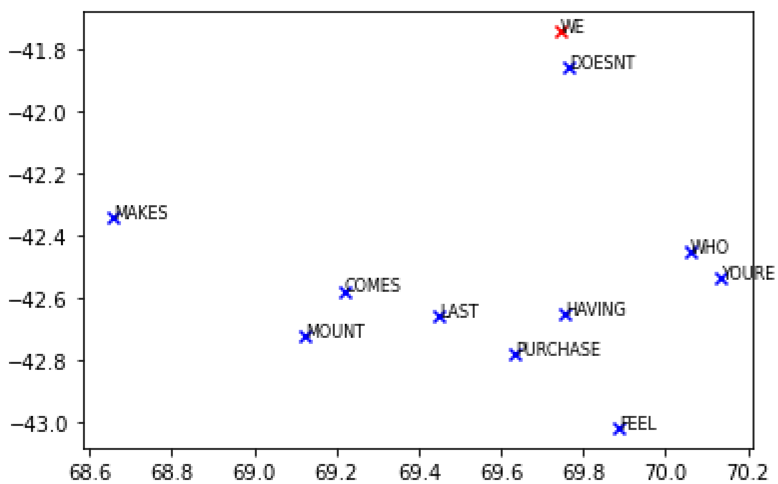
Negative sampling is a technique that is designed to prevent the issue of unrelated words ending up with positions close to each and reducing the total computation needed as well. The idea is to, instead of generating neighbour words when trying to find embeddings, make the output a 0 or 1 value that takes on a 1 if the two input words are close by. This difference might seem small but it brings about a large change in computation. One problem that arises though is that of completely unrelated words, far away words ending up as embeddings since the model never learns if it always outputs a 1. To fix this we randomly find words that are not close by and assign 0 so that the model can learn to handle zeroes. This technique is called Negative Sampling.

Question 2.3.1

Words picked for analysis, with their closest words and plots:

1. WE

```
0.01307973696384579 - 'DOESNT'

0.6011901133024367 - 'WHO'

0.7795662724092836 - 'YOURE'

0.8289999469416216 - 'HAVING'

0.9260837642941624 - 'LAST'

0.9756592804478714 - 'COMES'

1.0905602856073529 - 'PURCHASE'

1.3423632905032719 - 'MOUNT'

1.5424549724557437 - 'MAKES'

1.6491293097642483 - 'FEEL'
```



GPS

0.04144789098063484 - DESKTOP

0.06239287087373668 - FEELS

0.1208844208777009 - TEST

0.14863559359582723 - STAND

0.3553160531191679 - MODEM

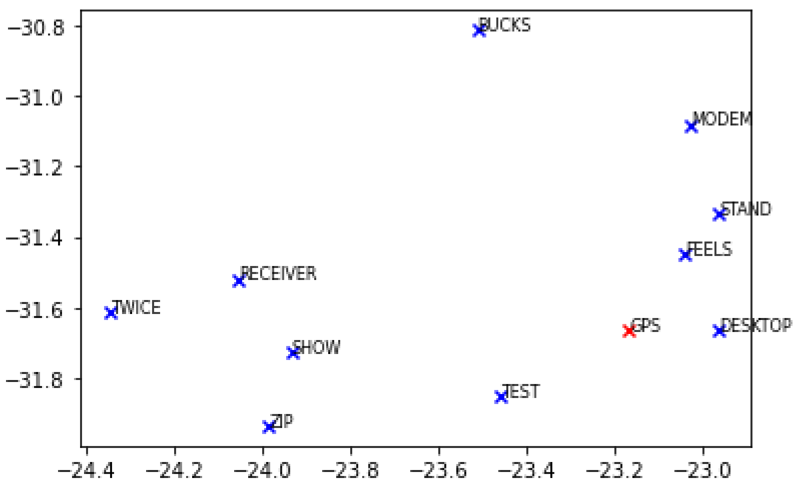
0.5890327049019106 - SHOW

0.7418820585371577 - ZIP

0.8080638312967494 - RECEIVER

0.8360376604287012 - BUCKS

1.3896798371933983 - TWICE



GOT

0.09178282263746951 - MAKE

0.12326241466507781 - DID

0.15544423880055547 - BEEN

0.1817957539606141 - COULD

0.24098053277703002 - LONG

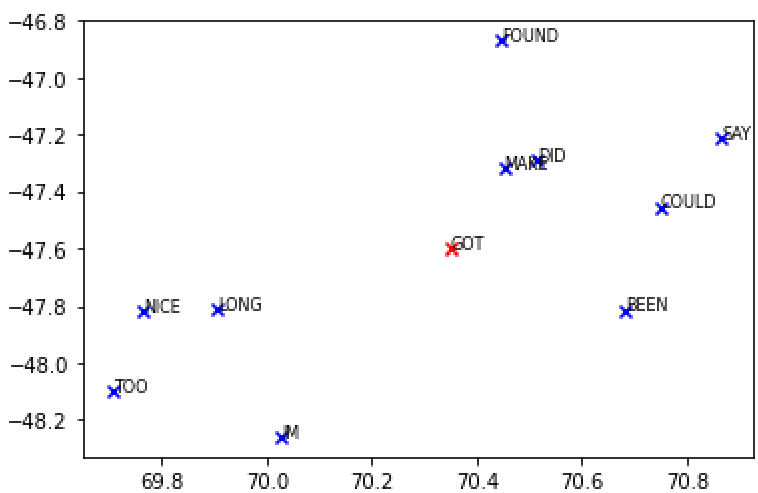
0.39024791738484055 - NICE

0.41957630253455136 - SAY

0.5362856539431959 - IM

0.5469142820802517 - FOUND

0.6620494654634967 - TOO



IS

0.014586753793992102 - OF

0.016879129339940846 - FOR

0.04185713693732396 - IN

0.046567063429392874 - I

0.04867319839831907 - ON

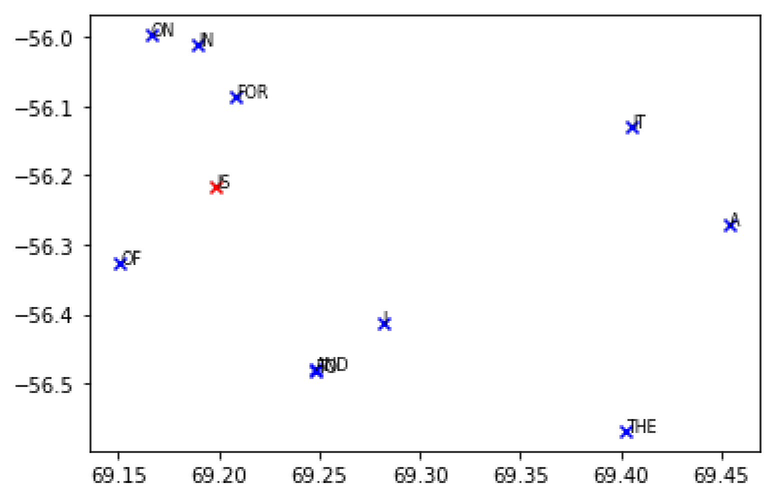
0.050067799747921526 - IT

0.06822394166374579 - A

0.07276496756821871 - AND

0.07416724001814146 - TO

0.1668169666227186 - THE



DECENT

0.14426669888780452 - FAST

0.2717830583314935 - HEAVY

0.4136160499911057 - PERFECTLY

0.4306442872803018 - TAPES

0.4714543001173297 - 10

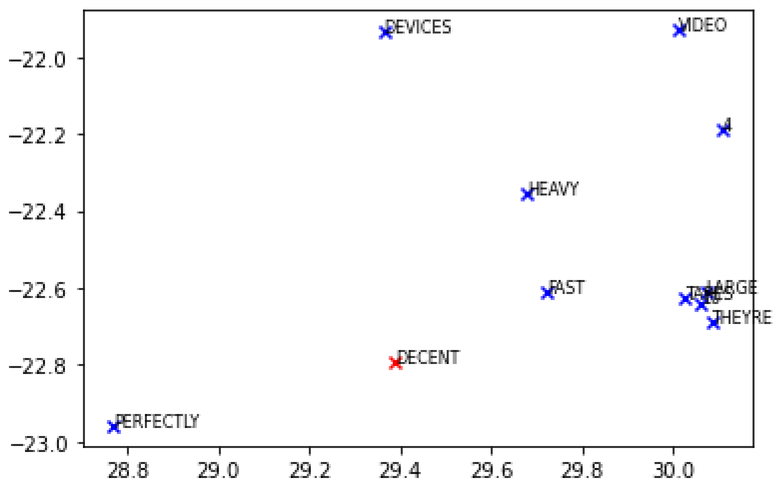
0.4934417255317385 - THEYRE

0.49794330995428027 - LARGE

0.7381343428060063 - DEVICES

0.8752976481810038 - 4

1.1295679938521062 - VIDEO



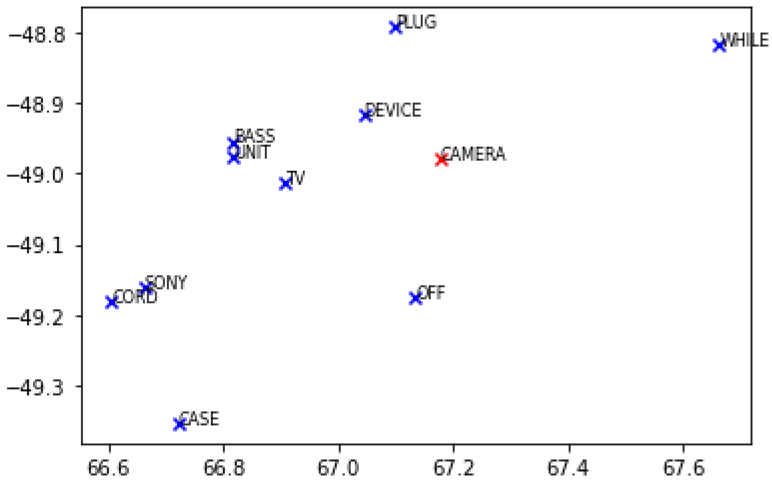
Question 2.3.2

The following are the results of Google's word2vec embeddings:

```
'microphone' - 0.5510498285293579

'flashguns' - 0.5686226487159729

'projector' - 0.5707793831825256

'Webcam - 0.5843442678451538

'webcam' - 0.5858358144760132

'viewfinder' - 0.596660852432251

'tripodd' - 0.6189838647842407

'Cameras' - 0.6350969076156616

'Camera' - 0.6848659515380859

'cameras' - 0.8131939768791199
```

And the following are the results from our embeddings:

```
0.021661442660843022 - DEVICE

0.03959543445671443 - OFF

0.04113973962375894 - PLUG

0.07354668476909865 - TV

0.12916748739371542 - UNIT

0.13128184858942404 - BASS

0.26206013055343647 - WHILE

0.296583566305344 - SONY

0.3462247927673161 - CASE

0.3683250044705346 - CORD
```



It seems Google's model has taken into account letter cases, leading to some repeated words. The words seem to be correct in our embedding as well as Google's but with some differences. Google's words are much more directly associated with cameras but ours also seem to be words that seem close.