Xinhao Liao
February 14th, 2020

SI630 Homework 2 – Word Embeddings

1 Task 1: Word2vec

The model is implemented in *word2vec_release.py* using the default parameter values.

2 Task 2: Save Your Outputs

The obtained vector embeddings are saved in *saved_W1.data* and *saved_W2.data* respectively.

3 Task 3: Word Similarities

The functions are also implemented in *word2vec_release.py*.

3.1 Problem 7 Most Similar Words.

When picking 10 target words "good", "bad", "food", "apple","tasteful","unbelievably","uncle","tool","tl
and "eat", the obtained neighbors from *get_neightbors(target_word)* are shown below.

Neighbors of "good":

word	similarity
good	1.0
great	0.5189637294469099
exceptional	0.5164033610889365
outstanding	0.4947771079410379
complaints	0.49453352779698956
poor	0.49382807612241675
awesome	0.49234368225211456
amazing	0.489580752743372
terrific	0.48538384075581187
bad	0.48147745350218885

Table 1: Top 10 words closet to 'good'

Neighbors of "bad":

word	similarity
bad	1.0
awful	0.5740880370900769
nasty	0.5685175393135137
horrid	0.560721380503272
terrible	0.5550031952565158
horrible	0.5540310315282168
lousy	0.5401260849517145
blah	0.5283637102433
so-so	0.5239793003622151
weird	0.5226704013645843

Table 2: Top 10 words closet to 'bad'

Neighbors of "food":

word	similarity
food	1.0
comfort	0.5311333165182095
junk	0.5048325642383873
foods	0.49642757458846243
thai	0.47768552501010675
indian	0.458046410700238
groceries	0.45689722027274415
preparedness	0.4547411545005067
homeless	0.4462811765054995
agricultural	0.44143478462963737

Table 3: Top 10 words closet to 'food'

Neighbors of "apple":

word	similarity
apple	1.0
apricot	0.5945317559739833
pineapple	0.5797147865619727
cider	0.574045414843073
orange	0.5631953112814618
pear	0.5545398484616841
watermelon	0.5517080552301864
carrot	0.5508883055993714
blackberry	0.5465732946795113
lemonade	0.5353339936431177

Table 4: Top 10 words closet to 'apple'

Neighbors of "tasteful":

word	similarity
tasteful	1.0
flavorable	0.6648998584787711
enjoyable	0.6102757583625451
satisfying	0.5886542357709932
flavourful	0.5337220770276547
exceptionally	0.5309873019712045
delightfully	0.5229512985419458
flavorful	0.5128002940533283
vey	0.49577751959220695
flavorfull	0.48900740491490136

Table 5: Top 10 words closet to 'tasteful'

Neighbors of "unbelievably":

word	similarity
unbelievably	1.0
incredibly	0.6560209895388734
amazingly	0.6459097533005582
deliciously	0.634295084308652
surprisingly	0.6077812546393908
ridiculously	0.5913476928194147
sooooo	0.5907824126741243
soooooo	0.587354169088698
outrageously	0.5865050671621082
downright	0.5843136327225497

Table 6: Top 10 words closet to 'unbelievably'

Neighbors of "uncle":

word	similarity
uncle	1.0
ben	0.6339233349029368
grandma	0.6007811225318777
aunt	0.5695072328686593
lee	0.5638886711341213
john	0.5526469447251433
henry	0.5520542334216348
mother	0.549000985871534
grandmother	0.5401059373435515
annie	0.5332676224900176

Table 7: Top 10 words closet to 'uncle'

Neighbors of "tool":

word	similarity
tool	1.0
clever	0.5227337824745387
functional	0.5019166448862813
device	0.4986355948379546
grip	0.4881801112427335
book	0.4779385612035798
points	0.4750315081443004
hygiene	0.4720801434741193
solution	0.4695596427449502
dispensing	0.46632184126811604

Table 8: Top 10 words closet to 'tool'

Neighbors of "think":

word	similarity
think	1.0
bet	0.599009844837198
doubt	0.5401689626044737
figured	0.5388053255391151
thought	0.5293429398437725
guess	0.5250417052127244
probably	0.5219838997109361
sure	0.507334199369886
alright	0.5059269431696788
personally	0.4993403811176592

Table 9: Top 10 words closet to 'think'

Neighbors of "eat":

word	similarity
eat	1.0
munch	0.5969995946703968
eaten	0.5819605675130636
eating	0.5436319777574993
consume	0.5390262245308862
dress	0.5370106904318279
indulge	0.5278677014007361
sneak	0.5026930770459025
eats	0.5026478242853435
handful	0.4989286870654517

Table 10: Top 10 words closet to 'eat'

As we can see, the obtained neighbors are generally semantically similar or relevant to the target words. And in most cases, the obtained neighbors are of the same part of speech as the target words. Yet we can also find that the similarities are generally not high enough, with most of the greatest ones lower than 0.6. This shows that the word vectors are not accurate enough, which is reasonable considering the small corpus used and the lack of training.

3.2 Problem 8 Word Analogy

Using the function *analogy([A,B,C])*, we expect to obtain word D where A is to B is like C is to D. Since the word embeddings are not good enough, the top 10 most closing words are returned instead of only 1. And we check the following 5 sets words.

analogy(["male","king","female"]) gives the following list, where "queen", the word we expect to obtain, is ranked No.5 in the list.

word	similarity
king	0.688780260406929
arthur	0.5003948986808116
sir	0.48995081249987815
francisco	0.43335811361163545
queen	0.4254647538594647
swiss	0.42071478757985104
udi	0.41785092317202244
lily	0.417570067638541
tully	0.4168002551078853
davidson	0.41168811478739675

analogy(["man","brother","woman"]) gives the following list, where "sister", the word we expect to obtain, is ranked No.3 in the list.

word	similarity
woman	0.7187219497123915
brother	0.6501705908633887
sister	0.6067261600774126
law	0.5993406260386167
friend	0.592124735046978
brother-in-law	0.5840260181019651
coworker	0.5765595709456082
co-worker	0.565554626831793
oldest	0.5653231057035278
sister-in-law	0.5363635800386253

analogy(["father","mother","grandfather"]) gives the following list, where "grandmother", the word we expect to obtain, is ranked No.3 in the list.

word	similarity
grandfather	0.8302620699087538
mother	0.6573450885352055
grandmother	0.5932980900477283
grandparents	0.57548948199355
parents	0.5018643300075586
girl	0.4993748494773893
sweden	0.4890925529309258
mum	0.4877746509876171
aunt	0.4806616136034274
grandkids	0.47927442871440906

analogy(["warm","warmer","cold"]) gives the following list, where "colder", the word we expect to obtain, is ranked No.3 in the list.

word	similarity
warmer	0.7533189876874306
cold	0.6846675962083107
colder	0.5618431086720828
cooler	0.531111510402407
weather	0.4992465752349333
auto	0.4460107121812116
monthly	0.4418651063259127
toddy	0.43362742462683745
summer	0.42478975688822773
shipped	0.4221026837416464

analogy(["small","big","short"]) gives the following list, where "long", the word we expect to obtain, is ranked No.3 in the list.

word	similarity
short	0.6377510501834618
big	0.4353373303841557
long	0.4342991760624615
assuming	0.4334105749655417
beyond	0.42544151440937283
assured	0.41999755140269324
loyal	0.4194412952152976
near	0.4061809767330886
mile	0.4053247081016448
biggest	0.3956485193060977

According to the results above, we can see that the word embeddings can generate the words with relationships in analogy, though with some error. A problem might be that vectors are currently too far away from each other in the cosine distances that the vectors are still close to the original vectors after some vector operations. The reason might be that the corpus is small, and the training is not enough.

4 Task 4: Intrinsic Evaluation: Word Similarity

The generated evaluation is given in *evaluation.csv* and uploaded to *Kaggle*. (Notice that due to a change of evaluation metrics on *Kaggle*, my current highest score on *Kaggle* was a result of the old evaluation metrics and not effective. I have chosen another submission for the final result, which should be the same as *evaluation.csv*.)