2015410084 컴퓨터학과 정윤진

2019-1R 자연어처리
이상근 교수님
2019.05.07 (재제출)

Assignment 3 - Wore2vec

Analysis of word analogy

A. Skip-Gram Negative Sampling

1. Options: Subsampled, dimension=64, epoch=1, learning rate=1

[Training Loss 결과]

```
# Training section (Subsampling)
emb,_ = word2vec_trainer(input_set, target_set, len(w2i), codedict, freqtable, mode=mode, NS=20, dimension=64, epoch=1,
```

```
# of training samples 46708010
Iteration: 1000000 Loss : 7.654623
Iteration: 2000000 Loss : 5.059421
Iteration: 3000000 Loss : 4.427601
Iteration: 4000000 Loss : 4.177590
Iteration: 5000000 Loss : 4.034789
Iteration: 6000000 Loss : 3.933675
Iteration: 7000000 Loss : 3.835700
Iteration: 8000000 Loss : 3.778920
Iteration: 9000000 Loss : 3.685837
Iteration: 10000000 Loss : 3.731983
Iteration: 11000000 Loss : 3.744131
Iteration: 12000000 Loss : 3.618265
Iteration: 13000000 Loss : 3.651451
Iteration: 14000000 Loss : 3.621797
Iteration: 15000000 Loss : 3.606777
Iteration: 16000000 Loss : 3.647783
Iteration: 17000000 Loss : 3.609555
Iteration: 18000000 Loss : 3.616155
Iteration: 19000000 Loss : 3.578858
Iteration: 20000000 Loss : 3.590594
Iteration: 21000000 Loss : 3.533950
Iteration: 22000000 Loss : 3.555097
Iteration: 23000000 Loss : 3.542762
Iteration: 24000000 Loss : 3.520544
Iteration: 25000000 Loss : 3.495258
Iteration: 26000000 Loss : 3.526438
Iteration: 27000000 Loss : 3.501265
Iteration: 28000000 Loss : 3.495406
Iteration: 29000000 Loss : 3.505055
Iteration: 30000000 Loss : 3.482589
Iteration: 31000000 Loss : 3.496674
Iteration: 32000000 Loss : 3.485552
Iteration: 33000000 Loss : 3.439194
Iteration: 34000000 Loss : 3.405046
Iteration: 35000000 Loss : 3.449131
Iteration: 36000000 Loss : 3.412199
Iteration: 37000000 Loss : 3.455745

Iteration: 38000000 Loss : 3.422245
Iteration: 39000000 Loss : 3.456103
Iteration: 40000000 Loss : 3.347547
Iteration: 41000000 Loss : 3.386247
Iteration: 42000000 Loss : 3.387384
Iteration: 43000000 Loss : 3.296019
Iteration: 44000000 Loss : 3.385767
Iteration: 45000000 Loss : 3.420860
Iteration: 46000000 Loss : 3.394718
```

Total Iteration= 46708010

Final Loss = 3.394718

[Analogical Word Testing 결과]

```
Analogical_Reasoning_Task(emb_dict)
```

```
 18200 iterations => ['machine', 'machines', 'car', 'cars'] car tensor(0.5292) tensor(0.9186)
 18300 iterations => ['melon', 'melons', 'woman', 'women'] woman tensor(0.5214) tensor(0.9215)
 18400 iterations => ['onion', 'onions', 'pig', 'pigs'] pig tensor(0.4882) tensor(0.8443)
 18500 iterations => ['pig', 'pigs', 'mango', 'mangoes'] pigs tensor(0.5552) tensor(0.7632)
 18600 iterations => ['road', 'roads', 'goat', 'goats'] goat tensor(0.5939) tensor(0.8403)
 Correct! ['woman', 'women', 'man', 'men'] men 93
 18700 iterations => ['decrease', 'decreases', 'provide', 'provides'] provide tensor(0.6418) tensor(0.8600)
 18800 iterations => ['enhance', 'enhances', 'write', 'writes'] write tensor(0.6054) tensor(0.8666)
 18900 iterations => ['go', 'goes', 'see', 'sees'] see tensor(0.6311) tensor(0.8555)
 19000 iterations => ['listen', 'listens', 'enhance', 'enhances'] impacts tensor(0.5949) tensor(0.8129)
 Correct! ['play', 'plays', 'say', 'says'] says 94
 19100 iterations => ['say', 'says', 'slow', 'slows'] slow tensor(0.6726) tensor(0.9006)
 19200 iterations => ['see', 'sees', 'go', 'goes'] go tensor(0.4992) tensor(0.8451)
 19300 iterations => ['slow', 'slows', 'think', 'thinks'] think tensor(0.5948) tensor(0.8598)
 19400 iterations => ['talk', 'talks', 'play', 'plays'] few tensor(0.6246) tensor(0.8167)
 19500 iterations => ['work', 'works', 'write', 'writes'] write tensor(0.6346) tensor(0.9176)
 Correct Answer:  94 when total # of words is  19558

 [Acuuracy]:  0.48062174046426015 [Words not in corpus]:  1717
```

Accuracy = 0.0048 (approximately)

Total number of Correct Answers = 94

Total number of Words not in corpus = 1717

2. Options: <u>Not</u> Subsampled, dimension=64, epoch=1, learning rate=1

[Training Loss 결과]

```
# Training section (No-Subsampling)
emb,_ = word2vec_trainer(input_set, target_set, len(w2i), codedict, freqtable, mode=mode, NS=20, dimension=64, epoch=1,
```

```
 # of training samples 167188440
 Iteration: 10000000 Loss : 3.750347
 Iteration: 20000000 Loss : 3.231344
 Iteration: 30000000 Loss : 3.183585
 Iteration: 40000000 Loss : 3.169564
 Iteration: 50000000 Loss : 3.130268
 Iteration: 60000000 Loss : 3.185463
 Iteration: 70000000 Loss : 3.157347
 Iteration: 80000000 Loss : 3.149769
 Iteration: 90000000 Loss : 3.118331
 Iteration: 100000000 Loss : 3.161987
 Iteration: 110000000 Loss : 3.166807
 Iteration: 120000000 Loss : 3.149533
```

Iteration이 반복되어도 Loss값이3.11 ~ 3.16 내에서 움직이는 것으로 보아 더이상 학습이 진행되지 않는다고 판단하였다. 무엇보다 Subsampling을 하지 않은 상태에서 Negative Sampling을 Skip-gram으로 실행시켰을 때 매우 큰 training sampl에 의해 실행시간이 굉장히 오래 걸렸다. 결국, Subsampling의 유무가 학습에 유의미한 변화를 주지는 않는다고 생각하여 Iteration 120000000 에서 중지하였다.

B. CBOW Negative Sampling.

1. Options: Subsampled, dimension=64, epoch=1, learning rate=1

[Training Loss 결과]

```
#CBOW (Subsampling)
emb,_ = word2vec_trainer(input_set, target_set, len(w2i), codedict, freqtable, mode=mode, NS=20, dimension=64, epoch=1,
```

```
# of training samples 4669773
Created Huffman Code Tree. The number of nodes:  71290
Iteration: 100000 Loss : 7.451588
Iteration: 200000 Loss : 4.999447
Iteration: 300000 Loss : 4.404350
Iteration: 400000 Loss : 4.234064
Iteration: 500000 Loss : 4.118116
Iteration: 600000 Loss : 4.048420
Iteration: 700000 Loss : 3.952099
Iteration: 800000 Loss : 3.915594
Iteration: 900000 Loss : 3.826313

Iteration: 3900000 Loss : 3.394928
Iteration: 4000000 Loss : 3.251652
Iteration: 4100000 Loss : 3.292549
Iteration: 4200000 Loss : 3.299085
Iteration: 4300000 Loss : 3.143159
Iteration: 4400000 Loss : 3.274124
Iteration: 4500000 Loss : 3.295920
Iteration: 4600000 Loss : 3.265339
```

Total Iteration= 4669773

Final Loss = 3.265339

[Analogical Word Testing 결과]

```
Analogical_Reasoning_Task(emb_dict)
```

```
18000 iterations => ['eye', 'eyes', 'dream', 'dreams'] dream tensor(0.2753) tensor(0.7822)
18100 iterations => ['hand', 'hands', 'computer', 'computers'] computer tensor(0.2147) tensor(0.8229)
18200 iterations => ['machine', 'machines', 'car', 'cars'] car tensor(0.2898) tensor(0.8050)
18300 iterations => ['melon', 'melons', 'woman', 'women'] woman tensor(0.1115) tensor(0.6173)
18400 iterations => ['onion', 'onions', 'pig', 'pigs'] onions tensor(0.1840) tensor(0.6315)
18500 iterations => ['pig', 'pigs', 'mango', 'mangoes'] kasavubu tensor(0.1534) tensor(0.5210)
18600 iterations => ['road', 'roads', 'goat', 'goats'] goat tensor(0.1079) tensor(0.6712)
18700 iterations => ['decrease', 'decreases', 'provide', 'provides'] decreases tensor(0.2737) tensor(0.7707)
18800 iterations => ['enhance', 'enhances', 'write', 'writes'] write tensor(0.1447) tensor(0.7198)
18900 iterations => ['go', 'goes', 'see', 'sees'] see tensor(0.0621) tensor(0.7450)
19000 iterations => ['listen', 'listens', 'enhance', 'enhances'] enhance tensor(-0.0854) tensor(0.6252)
19100 iterations => ['say', 'says', 'slow', 'slows'] slow tensor(0.3209) tensor(0.7499)
19200 iterations => ['see', 'sees', 'go', 'goes'] go tensor(0.2263) tensor(0.7575)
19300 iterations => ['slow', 'slows', 'think', 'thinks'] think tensor(0.1876) tensor(0.6908)
19400 iterations => ['talk', 'talks', 'play', 'plays'] play tensor(0.0294) tensor(0.6714)
19500 iterations => ['work', 'works', 'write', 'writes'] write tensor(0.1563) tensor(0.8194)
Correct Answer:  11 when total # of words is  19558

[Acuuracy]:  0.05624296962879641 [Words not in corpus]:  1717
```

Accuracy = 0.00056 (approximately)

Total number of Correct Answers = 11

Total number of Words not in corpus = 1717

앞선 Skip-gram보다 최종 Loss값은 근소한 차이로 작았으나 실제 Analogical Testing을 한 결과 Skip-gram에 비해 그 정확성이 현저히 떨어짐을 확인하였다. 즉, CBOW는 Skip-gram에 비해 트레이닝 데이터내에서는 비슷한 성과를 거둘 수 있지만 실제 비슷한 단어의 벡터들을 인식함에 있어서는 학습이 잘 이루어지지 않았다. 이를 통해 두 학습에 분명한 차이가 있음을 알 수 있었다. Subsampling을 하였기에 Corpus에 없는 단어의 수는 1717개로 같았다.

더 많은 옵션으로 돌려보고 싶었으나 시간이 오래 걸리고 중간에 그만 둘 수 없다는 여러 한계때문에 위와 같은 세 옵션을 이상으로 보고서를 마칩니다.