环境配置

1. 在windows下，即使用mingw32-make编译项目，wordvec2.c中的关键字CLOCKS_PER_SEC等关键字也没有定义；

- 把source.archive.zip放到Linux远程服务器，unzip

2. cd trunk; make; 完成项目编译

   ~~带~~和

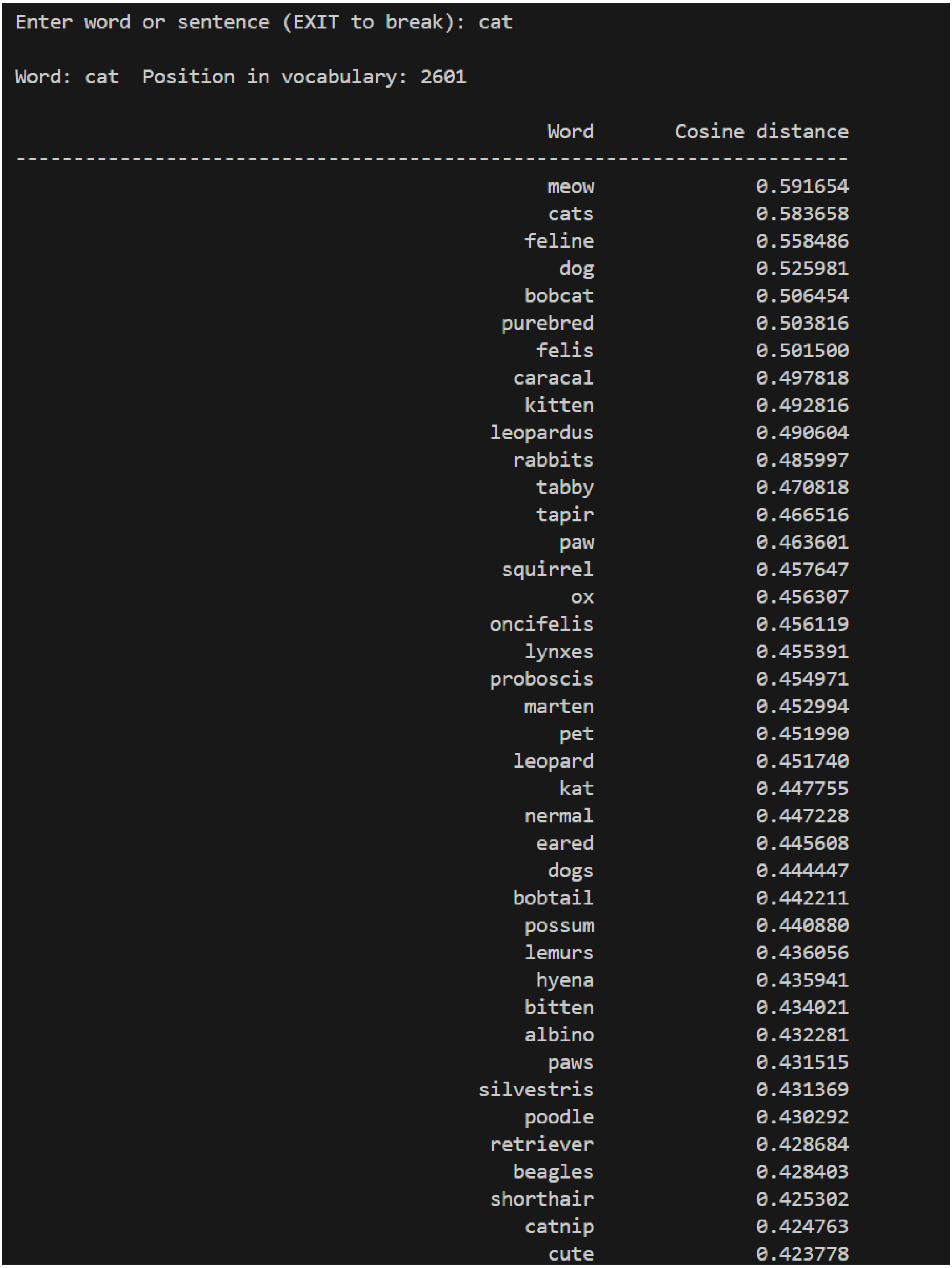
训练vector

1. CBOW+negative sample(=25)

参数设置

```
time ./word2vec -train text8 -output vectors.bin -cbow 1 -size 200 -window 8 -negative 25 -hs 0 -sample 1e-4 -threads 20 -binary 1 -iter 15
```

例子



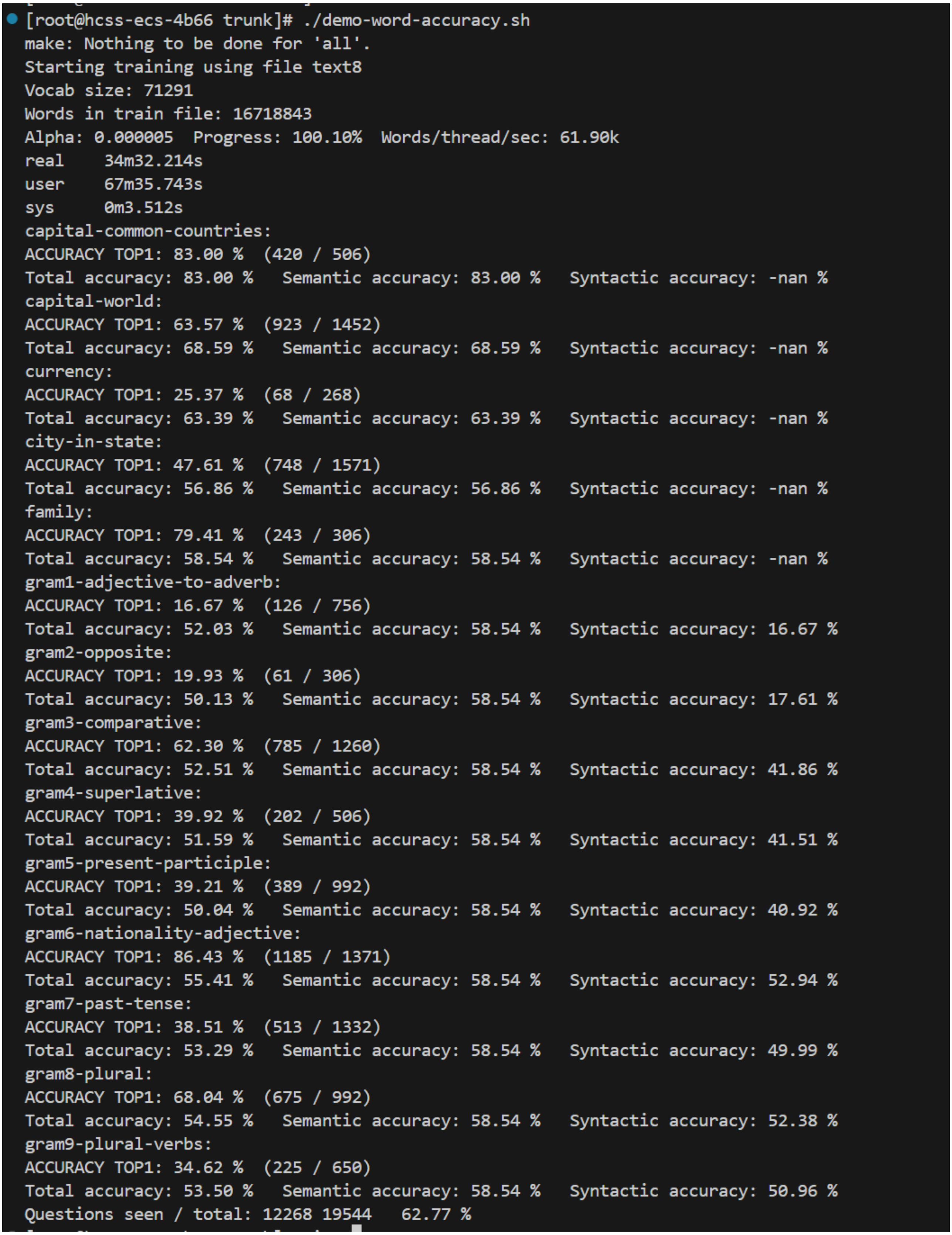
```
Enter word or sentence (EXIT to break): cat

Word: cat  Position in vocabulary: 2601

                                              Word       Cosine distance
------------------------------------------------------------------------
                                              meow              0.591654
                                              cats              0.583658
                                            feline              0.558486
                                               dog              0.525981
                                            bobcat              0.506454
                                          purebred              0.503816
                                             felis              0.501500
                                           caracal              0.497818
                                            kitten              0.492816
                                         leopardus              0.490604
                                           rabbits              0.485997
                                             tabby              0.470818
                                             tapir              0.466516
                                               paw              0.463601
                                          squirrel              0.457647
                                                ox              0.456307
                                         oncifelis              0.456119
                                            lynxes              0.455391
                                         proboscis              0.454971
                                            marten              0.452994
                                               pet              0.451990
                                           leopard              0.451740
                                               kat              0.447755
                                            nermal              0.447228
                                             eared              0.445608
                                              dogs              0.444447
                                           bobtail              0.442211
                                            possum              0.440880
                                            lemurs              0.436056
                                             hyena              0.435941
                                            bitten              0.434021
                                            albino              0.432281
                                              paws              0.431515
                                        silvestris              0.431369
                                            poodle              0.430292
                                         retriever              0.428684
                                           beagles              0.428403
                                         shorthair              0.425302
                                            catnip              0.424763
                                              cute              0.423778
```

单词准确率

- ./demo-word.sh



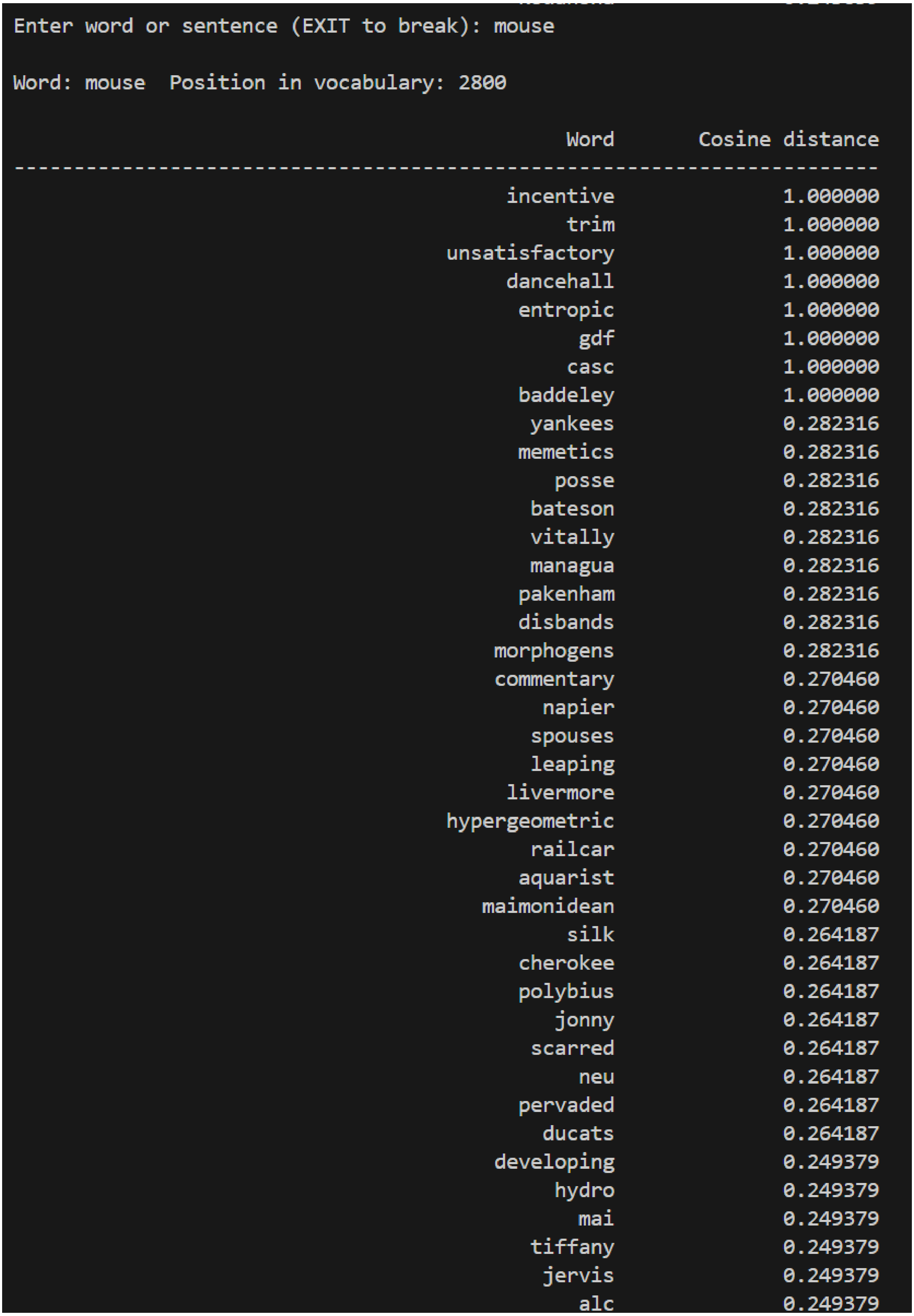
```
[root@hcss-ecs-4b66 trunk]# ./demo-word-accuracy.sh
make: Nothing to be done for 'all'.
Starting training using file text8
Vocab size: 71291
Words in train file: 16718843
Alpha: 0.000005  Progress: 100.10%  Words/thread/sec: 61.90k
real    34m32.214s
user    67m35.743s
sys     0m3.512s
capital-common-countries:
ACCURACY TOP1: 83.00 %  (420 / 506)
Total accuracy: 83.00 %   Semantic accuracy: 83.00 %   Syntactic accuracy: -nan %
capital-world:
ACCURACY TOP1: 63.57 %  (923 / 1452)
Total accuracy: 68.59 %   Semantic accuracy: 68.59 %   Syntactic accuracy: -nan %
currency:
ACCURACY TOP1: 25.37 %  (68 / 268)
Total accuracy: 63.39 %   Semantic accuracy: 63.39 %   Syntactic accuracy: -nan %
city-in-state:
ACCURACY TOP1: 47.61 %  (748 / 1571)
Total accuracy: 56.86 %   Semantic accuracy: 56.86 %   Syntactic accuracy: -nan %
family:
ACCURACY TOP1: 79.41 %  (243 / 306)
Total accuracy: 58.54 %   Semantic accuracy: 58.54 %   Syntactic accuracy: -nan %
gram1-adjective-to-adverb:
ACCURACY TOP1: 16.67 %  (126 / 756)
Total accuracy: 52.03 %   Semantic accuracy: 58.54 %   Syntactic accuracy: 16.67 %
gram2-opposite:
ACCURACY TOP1: 19.93 %  (61 / 306)
Total accuracy: 50.13 %   Semantic accuracy: 58.54 %   Syntactic accuracy: 17.61 %
gram3-comparative:
ACCURACY TOP1: 62.30 %  (785 / 1260)
Total accuracy: 52.51 %   Semantic accuracy: 58.54 %   Syntactic accuracy: 41.86 %
gram4-superlative:
ACCURACY TOP1: 39.92 %  (202 / 506)
Total accuracy: 51.59 %   Semantic accuracy: 58.54 %   Syntactic accuracy: 41.51 %
gram5-present-participle:
ACCURACY TOP1: 39.21 %  (389 / 992)
Total accuracy: 50.04 %   Semantic accuracy: 58.54 %   Syntactic accuracy: 40.92 %
gram6-nationality-adjective:
ACCURACY TOP1: 86.43 %  (1185 / 1371)
Total accuracy: 55.41 %   Semantic accuracy: 58.54 %   Syntactic accuracy: 52.94 %
gram7-past-tense:
ACCURACY TOP1: 38.51 %  (513 / 1332)
Total accuracy: 53.29 %   Semantic accuracy: 58.54 %   Syntactic accuracy: 49.99 %
gram8-plural:
ACCURACY TOP1: 68.04 %  (675 / 992)
Total accuracy: 54.55 %   Semantic accuracy: 58.54 %   Syntactic accuracy: 52.38 %
gram9-plural-verbs:
ACCURACY TOP1: 34.62 %  (225 / 650)
Total accuracy: 53.50 %   Semantic accuracy: 58.54 %   Syntactic accuracy: 50.96 %
Questions seen / total: 12268 19544   62.77 %
```

2. CBOW+hs

参数设置

```
time ./word2vec -train text8 -output vectors.bin -cbow 1 -size 200 -window 8 -negative 0 -hs 1 -sample 1e-4 -threads 20 -binary 1 -iter 15
```

例子



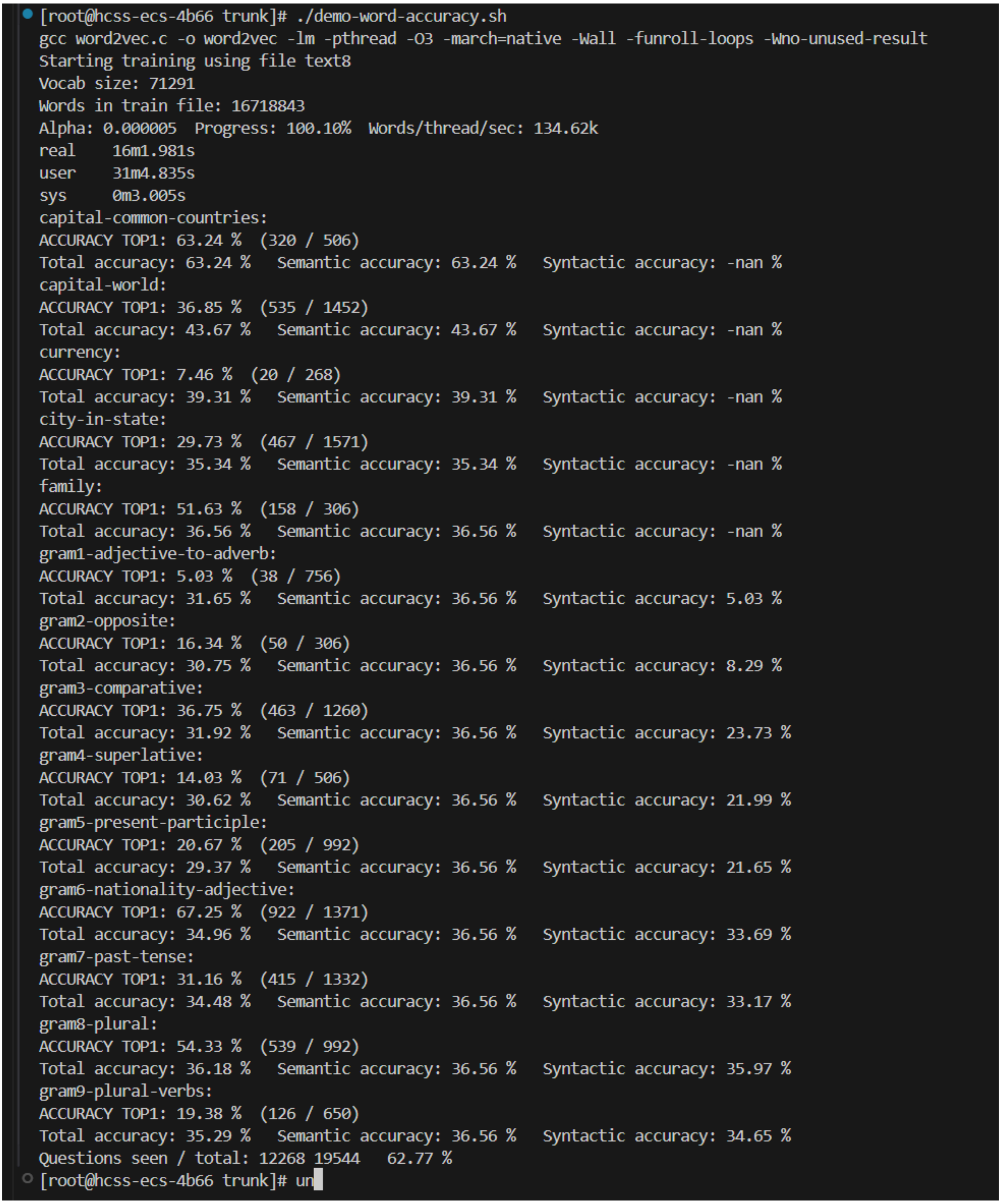
```
Enter word or sentence (EXIT to break): mouse

Word: mouse  Position in vocabulary: 2800

                                              Word       Cosine distance
------------------------------------------------------------------------
                                         incentive              1.000000
                                              trim              1.000000
                                    unsatisfactory              1.000000
                                         dancehall              1.000000
                                          entropic              1.000000
                                               gdf              1.000000
                                              casc              1.000000
                                          baddeley              1.000000
                                           yankees              0.282316
                                          memetics              0.282316
                                             posse              0.282316
                                           bateson              0.282316
                                           vitally              0.282316
                                           managua              0.282316
                                          pakenham              0.282316
                                          disbands              0.282316
                                        morphogens              0.282316
                                        commentary              0.270460
                                            napier              0.270460
                                           spouses              0.270460
                                           leaping              0.270460
                                         livermore              0.270460
                                    hypergeometric              0.270460
                                           railcar              0.270460
                                          aquarist              0.270460
                                       maimonidean              0.270460
                                              silk              0.264187
                                          cherokee              0.264187
                                          polybius              0.264187
                                             jonny              0.264187
                                           scarred              0.264187
                                               neu              0.264187
                                          pervaded              0.264187
                                            ducats              0.264187
                                        developing              0.249379
                                             hydro              0.249379
                                               mai              0.249379
                                           tiffany              0.249379
                                            jervis              0.249379
                                               alc              0.249379
```

- 将老鼠与刺激性、苗条的相关联

单词准确率



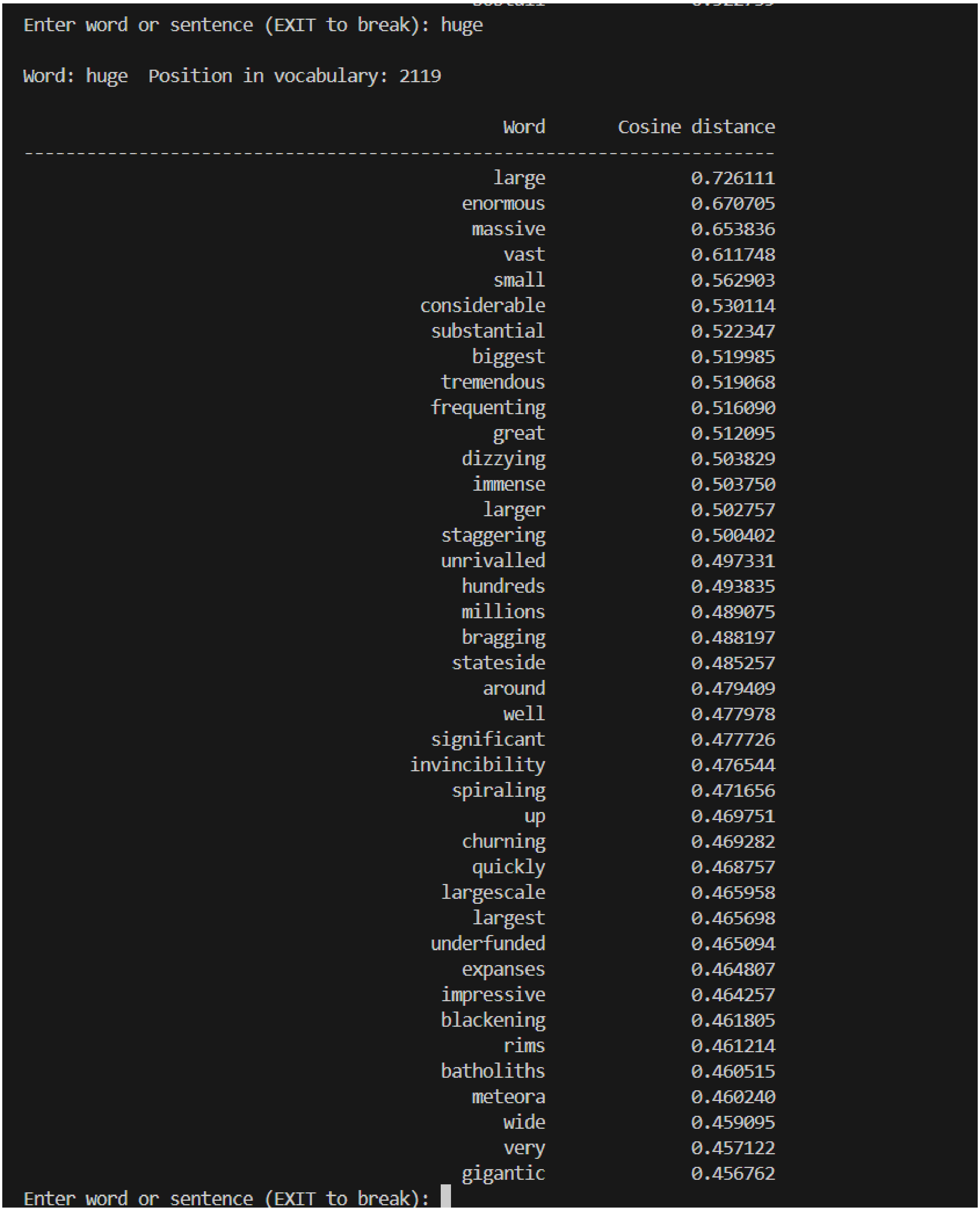
```
[root@hcss-ecs-4b66 trunk]# ./demo-word-accuracy.sh
gcc word2vec.c -o word2vec -lm -pthread -O3 -march=native -Wall -funroll-loops -Wno-unused-result
Starting training using file text8
Vocab size: 71291
Words in train file: 16718843
Alpha: 0.000005  Progress: 100.10%  Words/thread/sec: 134.62k
real    16m1.981s
user    31m4.835s
sys     0m3.005s
capital-common-countries:
ACCURACY TOP1: 63.24 %  (320 / 506)
Total accuracy: 63.24 %   Semantic accuracy: 63.24 %   Syntactic accuracy: -nan %
capital-world:
ACCURACY TOP1: 36.85 %  (535 / 1452)
Total accuracy: 43.67 %   Semantic accuracy: 43.67 %   Syntactic accuracy: -nan %
currency:
ACCURACY TOP1: 7.46 %  (20 / 268)
Total accuracy: 39.31 %   Semantic accuracy: 39.31 %   Syntactic accuracy: -nan %
city-in-state:
ACCURACY TOP1: 29.73 %  (467 / 1571)
Total accuracy: 35.34 %   Semantic accuracy: 35.34 %   Syntactic accuracy: -nan %
family:
ACCURACY TOP1: 51.63 %  (158 / 306)
Total accuracy: 36.56 %   Semantic accuracy: 36.56 %   Syntactic accuracy: -nan %
gram1-adjective-to-adverb:
ACCURACY TOP1: 5.03 %  (38 / 756)
Total accuracy: 31.65 %   Semantic accuracy: 36.56 %   Syntactic accuracy: 5.03 %
gram2-opposite:
ACCURACY TOP1: 16.34 %  (50 / 306)
Total accuracy: 30.75 %   Semantic accuracy: 36.56 %   Syntactic accuracy: 8.29 %
gram3-comparative:
ACCURACY TOP1: 36.75 %  (463 / 1260)
Total accuracy: 31.92 %   Semantic accuracy: 36.56 %   Syntactic accuracy: 23.73 %
gram4-superlative:
ACCURACY TOP1: 14.03 %  (71 / 506)
Total accuracy: 30.62 %   Semantic accuracy: 36.56 %   Syntactic accuracy: 21.99 %
gram5-present-participle:
ACCURACY TOP1: 20.67 %  (205 / 992)
Total accuracy: 29.37 %   Semantic accuracy: 36.56 %   Syntactic accuracy: 21.65 %
gram6-nationality-adjective:
ACCURACY TOP1: 67.25 %  (922 / 1371)
Total accuracy: 34.96 %   Semantic accuracy: 36.56 %   Syntactic accuracy: 33.69 %
gram7-past-tense:
ACCURACY TOP1: 31.16 %  (415 / 1332)
Total accuracy: 34.48 %   Semantic accuracy: 36.56 %   Syntactic accuracy: 33.17 %
gram8-plural:
ACCURACY TOP1: 54.33 %  (539 / 992)
Total accuracy: 36.18 %   Semantic accuracy: 36.56 %   Syntactic accuracy: 35.97 %
gram9-plural-verbs:
ACCURACY TOP1: 19.38 %  (126 / 650)
Total accuracy: 35.29 %   Semantic accuracy: 36.56 %   Syntactic accuracy: 34.65 %
Questions seen / total: 12268 19544   62.77 %
[root@hcss-ecs-4b66 trunk]# un
```

3. SG+negetive sample(=25)

- 用时：2024年4月23日19:03:56——2024年4月23日21:30:06
- 参数设置：

```
time ./word2vec -train text8 -output vectors.bin -cbow 0 -size 200 -window 8 -negative 25 -hs 0 -sample 1e-4 -threads 20 -binary 1 -iter 15
```

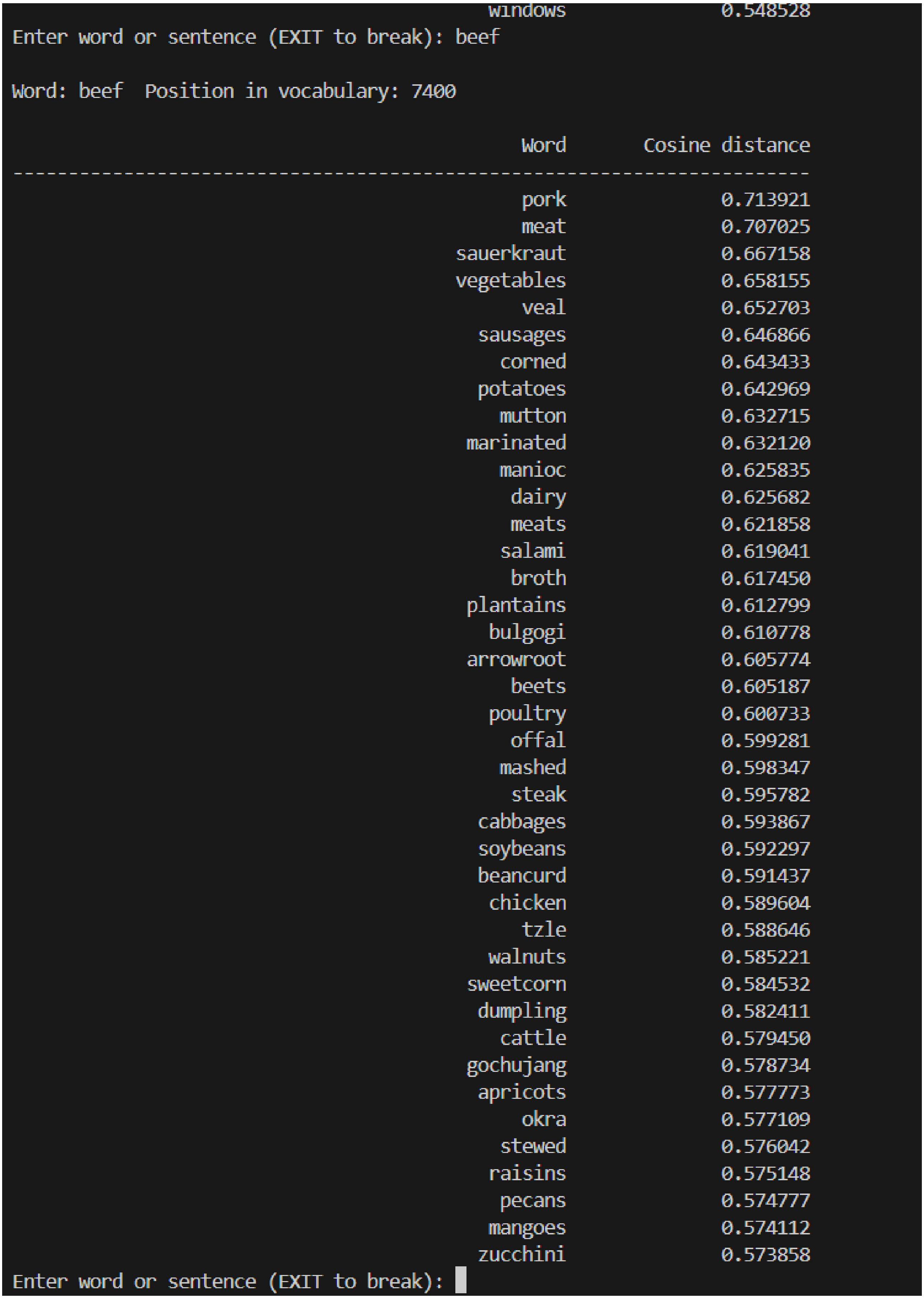
例子



```
Enter word or sentence (EXIT to break): huge

Word: huge  Position in vocabulary: 2119

                                              Word       Cosine distance
------------------------------------------------------------------------
                                             large              0.726111
                                          enormous              0.670705
                                           massive              0.653836
                                              vast              0.611748
                                             small              0.562903
                                      considerable              0.530114
                                       substantial              0.522347
                                           biggest              0.519985
                                        tremendous              0.519068
                                       frequenting              0.516090
                                             great              0.512095
                                          dizzying              0.503829
                                           immense              0.503750
                                            larger              0.502757
                                        staggering              0.500402
                                        unrivalled              0.497331
                                          hundreds              0.493835
                                          millions              0.489075
                                          bragging              0.488197
                                         stateside              0.485257
                                            around              0.479409
                                              well              0.477978
                                       significant              0.477726
                                     invincibility              0.476544
                                         spiraling              0.471656
                                                up              0.469751
                                          churning              0.469282
                                           quickly              0.468757
                                        largescale              0.465958
                                           largest              0.465698
                                       underfunded              0.465094
                                          expanses              0.464807
                                        impressive              0.464257
                                        blackening              0.461805
                                              rims              0.461214
                                        batholiths              0.460515
                                           meteora              0.460240
                                              wide              0.459095
                                              very              0.457122
                                          gigantic              0.456762
Enter word or sentence (EXIT to break):
```

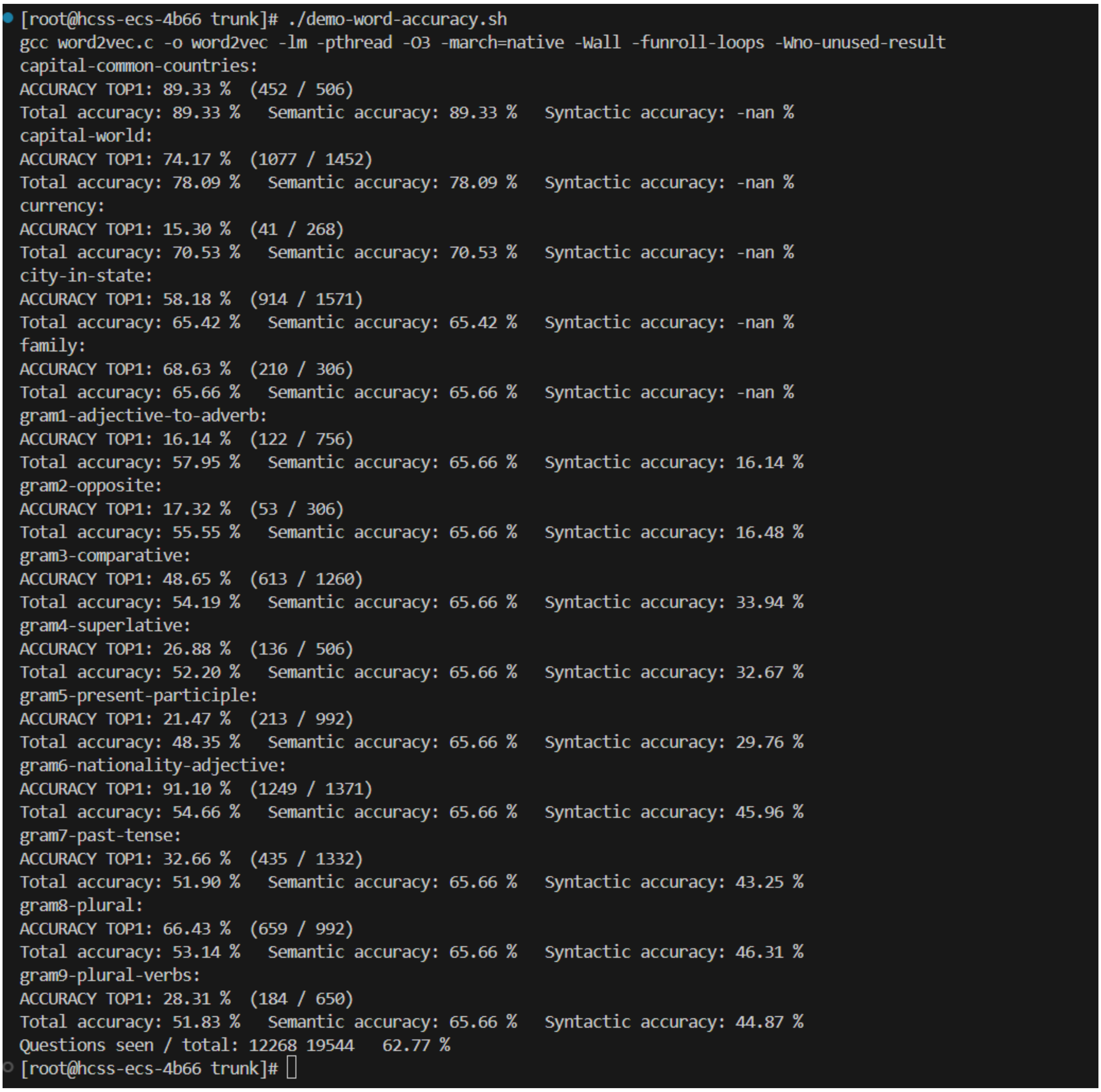


```
                                           windows              0.548528
Enter word or sentence (EXIT to break): beef

Word: beef  Position in vocabulary: 7400

                                              Word       Cosine distance
------------------------------------------------------------------------
                                              pork              0.713921
                                              meat              0.707025
                                        sauerkraut              0.667158
                                        vegetables              0.658155
                                              veal              0.652703
                                          sausages              0.646866
                                            corned              0.643433
                                          potatoes              0.642969
                                            mutton              0.632715
                                         marinated              0.632120
                                            manioc              0.625835
                                             dairy              0.625682
                                             meats              0.621858
                                            salami              0.619041
                                             broth              0.617450
                                         plantains              0.612799
                                           bulgogi              0.610778
                                         arrowroot              0.605774
                                             beets              0.605187
                                           poultry              0.600733
                                             offal              0.599281
                                            mashed              0.598347
                                             steak              0.595782
                                          cabbages              0.593867
                                          soybeans              0.592297
                                          beancurd              0.591437
                                           chicken              0.589604
                                              tzle              0.588646
                                           walnuts              0.585221
                                         sweetcorn              0.584532
                                          dumpling              0.582411
                                            cattle              0.579450
                                         gochujang              0.578734
                                          apricots              0.577773
                                              okra              0.577109
                                            stewed              0.576042
                                           raisins              0.575148
                                            pecans              0.574777
                                           mangoes              0.574112
                                          zucchini              0.573858
Enter word or sentence (EXIT to break): 
```

准确率



```
[root@hcss-ecs-4b66 trunk]# ./demo-word-accuracy.sh
gcc word2vec.c -o word2vec -lm -pthread -O3 -march=native -Wall -funroll-loops -Wno-unused-result
capital-common-countries:
ACCURACY TOP1: 89.33 %  (452 / 506)
Total accuracy: 89.33 %   Semantic accuracy: 89.33 %   Syntactic accuracy: -nan %
capital-world:
ACCURACY TOP1: 74.17 %  (1077 / 1452)
Total accuracy: 78.09 %   Semantic accuracy: 78.09 %   Syntactic accuracy: -nan %
currency:
ACCURACY TOP1: 15.30 %  (41 / 268)
Total accuracy: 70.53 %   Semantic accuracy: 70.53 %   Syntactic accuracy: -nan %
city-in-state:
ACCURACY TOP1: 58.18 %  (914 / 1571)
Total accuracy: 65.42 %   Semantic accuracy: 65.42 %   Syntactic accuracy: -nan %
family:
ACCURACY TOP1: 68.63 %  (210 / 306)
Total accuracy: 65.66 %   Semantic accuracy: 65.66 %   Syntactic accuracy: -nan %
gram1-adjective-to-adverb:
ACCURACY TOP1: 16.14 %  (122 / 756)
Total accuracy: 57.95 %   Semantic accuracy: 65.66 %   Syntactic accuracy: 16.14 %
gram2-opposite:
ACCURACY TOP1: 17.32 %  (53 / 306)
Total accuracy: 55.55 %   Semantic accuracy: 65.66 %   Syntactic accuracy: 16.48 %
gram3-comparative:
ACCURACY TOP1: 48.65 %  (613 / 1260)
Total accuracy: 54.19 %   Semantic accuracy: 65.66 %   Syntactic accuracy: 33.94 %
gram4-superlative:
ACCURACY TOP1: 26.88 %  (136 / 506)
Total accuracy: 52.20 %   Semantic accuracy: 65.66 %   Syntactic accuracy: 32.67 %
gram5-present-participle:
ACCURACY TOP1: 21.47 %  (213 / 992)
Total accuracy: 48.35 %   Semantic accuracy: 65.66 %   Syntactic accuracy: 29.76 %
gram6-nationality-adjective:
ACCURACY TOP1: 91.10 %  (1249 / 1371)
Total accuracy: 54.66 %   Semantic accuracy: 65.66 %   Syntactic accuracy: 45.96 %
gram7-past-tense:
ACCURACY TOP1: 32.66 %  (435 / 1332)
Total accuracy: 51.90 %   Semantic accuracy: 65.66 %   Syntactic accuracy: 43.25 %
gram8-plural:
ACCURACY TOP1: 66.43 %  (659 / 992)
Total accuracy: 53.14 %   Semantic accuracy: 65.66 %   Syntactic accuracy: 46.31 %
gram9-plural-verbs:
ACCURACY TOP1: 28.31 %  (184 / 650)
Total accuracy: 51.83 %   Semantic accuracy: 65.66 %   Syntactic accuracy: 44.87 %
Questions seen / total: 12268 19544   62.77 %
[root@hcss-ecs-4b66 trunk]# 
```

4. SG+hs

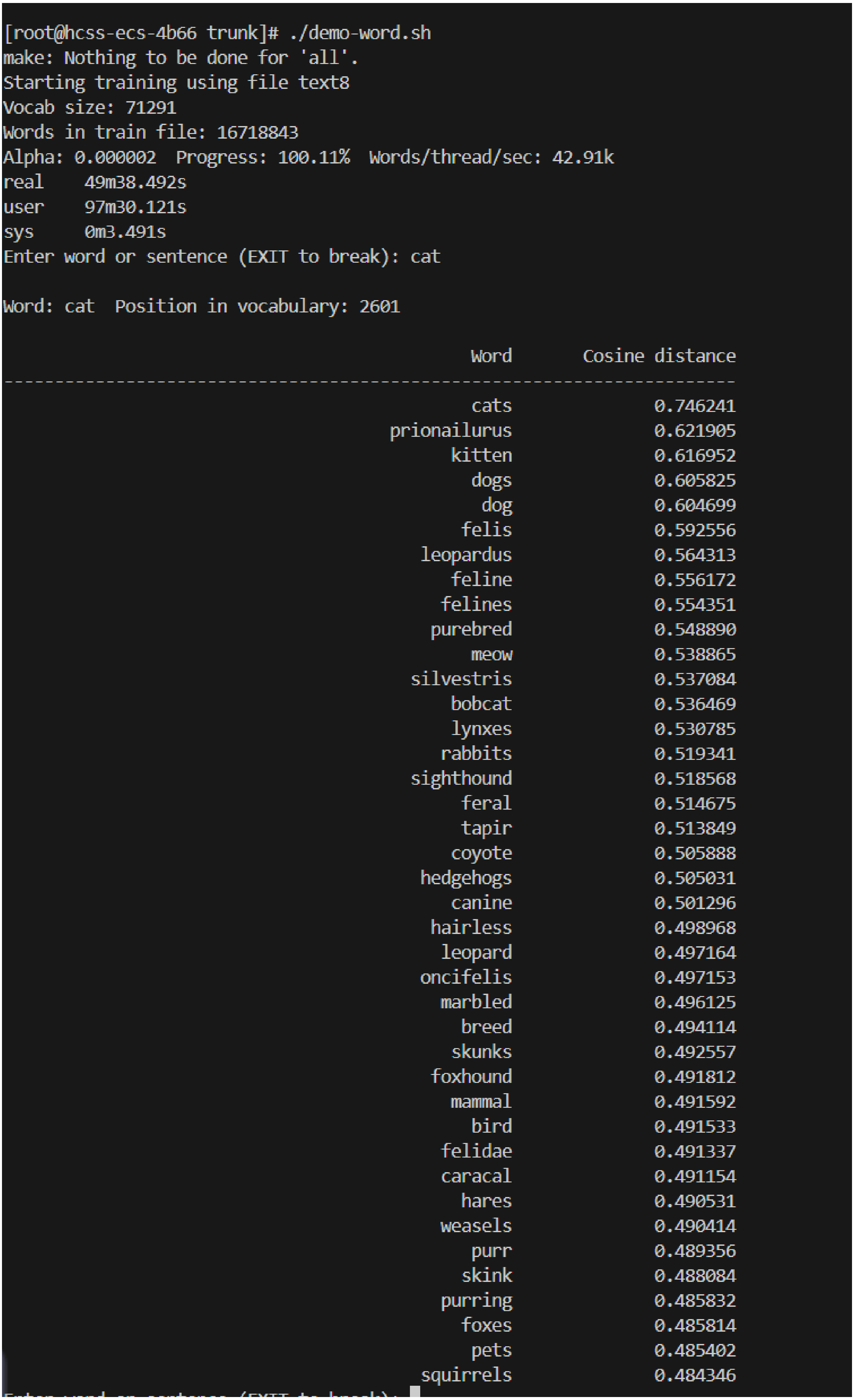
用时：1h

参数设置

```
time ./word2vec -train text8 -output vectors.bin -cbow 0 -size 200 -window 8 -negative 0 -hs 1 -sample 1e-4 -threads 20 -binary 1 -iter 15
```

例子

- cat



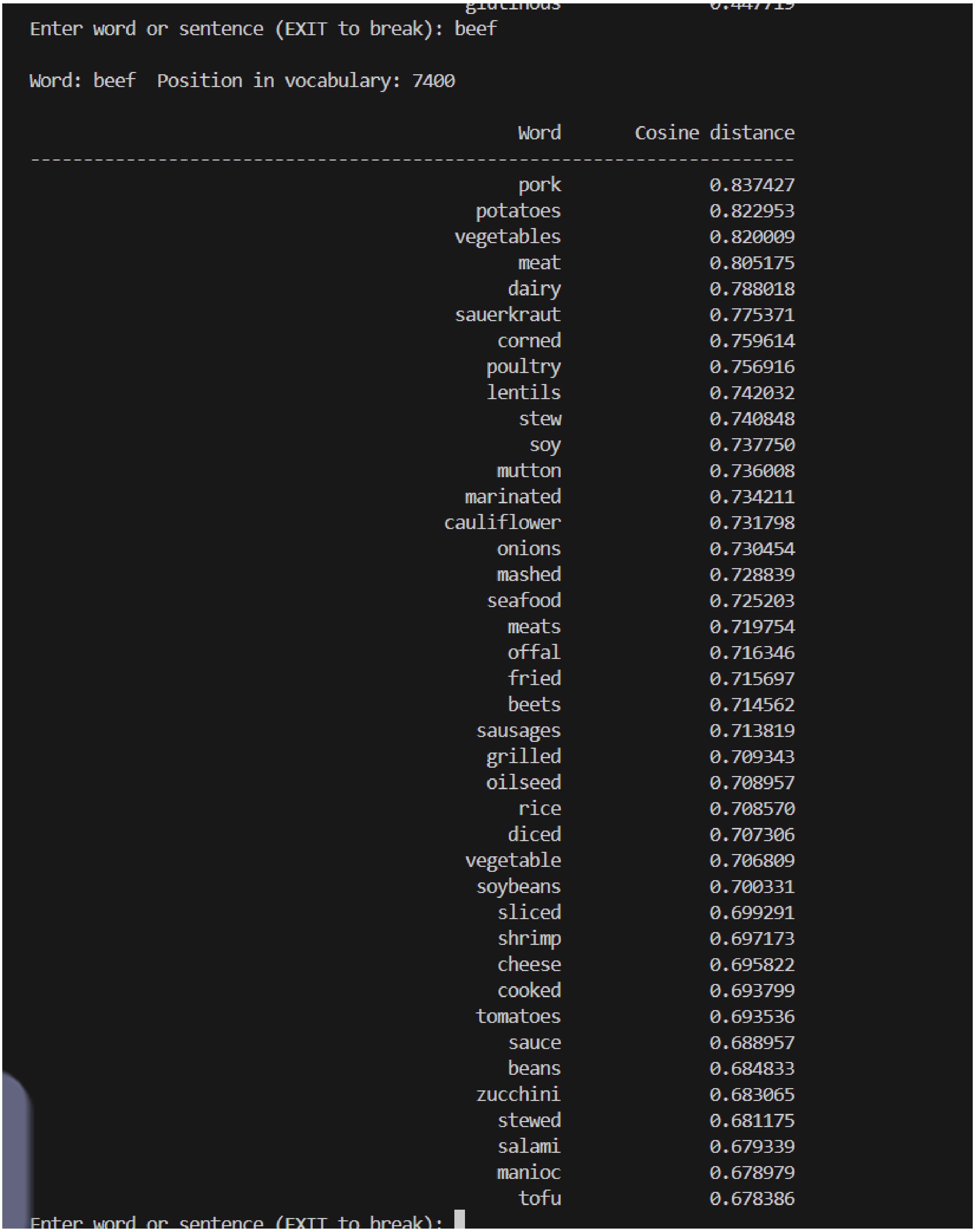
```
[root@hcss-ecs-4b66 trunk]# ./demo-word.sh
make: Nothing to be done for 'all'.
Starting training using file text8
Vocab size: 71291
Words in train file: 16718843
Alpha: 0.000002  Progress: 100.11%  Words/thread/sec: 42.91k
real    49m38.492s
user    97m30.121s
sys     0m3.491s
Enter word or sentence (EXIT to break): cat

Word: cat  Position in vocabulary: 2601

                                              Word       Cosine distance
------------------------------------------------------------------------
                                              cats              0.746241
                                      prionailurus              0.621905
                                            kitten              0.616952
                                              dogs              0.605825
                                               dog              0.604699
                                             felis              0.592556
                                         leopardus              0.564313
                                            feline              0.556172
                                           felines              0.554351
                                          purebred              0.548890
                                              meow              0.538865
                                        silvestris              0.537084
                                            bobcat              0.536469
                                            lynxes              0.530785
                                           rabbits              0.519341
                                        sighthound              0.518568
                                             feral              0.514675
                                             tapir              0.513849
                                            coyote              0.505888
                                         hedgehogs              0.505031
                                            canine              0.501296
                                          hairless              0.498968
                                           leopard              0.497164
                                         oncifelis              0.497153
                                           marbled              0.496125
                                             breed              0.494114
                                            skunks              0.492557
                                          foxhound              0.491812
                                            mammal              0.491592
                                              bird              0.491533
                                           felidae              0.491337
                                           caracal              0.491154
                                             hares              0.490531
                                           weasels              0.490414
                                              purr              0.489356
                                             skink              0.488084
                                           purring              0.485832
                                             foxes              0.485814
                                              pets              0.485402
                                         squirrels              0.484346
Enter word or sentence (EXIT to break): 
```

```
Enter word or sentence (EXIT to break):
```

- beef

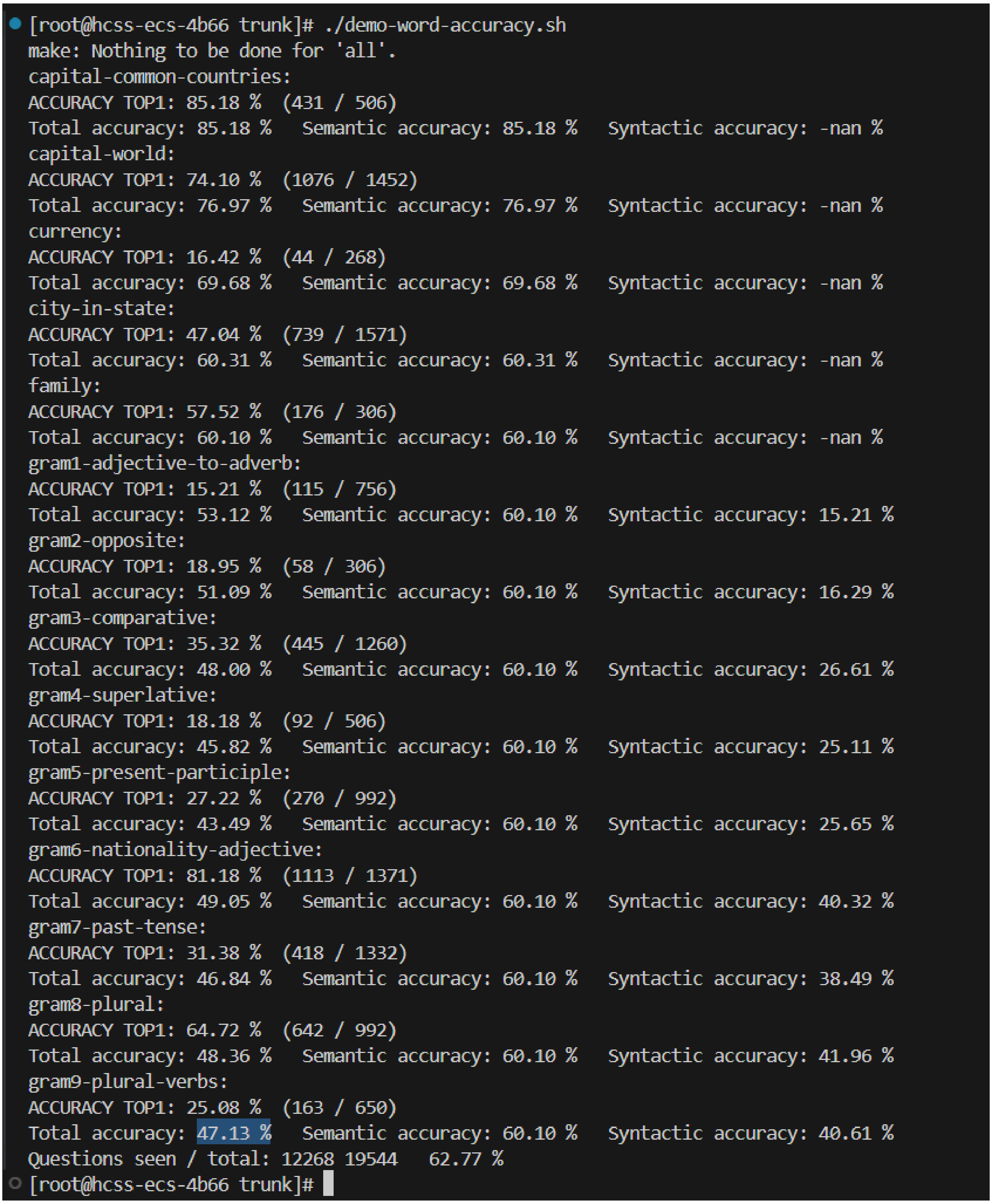


```
                                         glutinous              0.447719
Enter word or sentence (EXIT to break): beef

Word: beef  Position in vocabulary: 7400

                                              Word       Cosine distance
------------------------------------------------------------------------
                                              pork              0.837427
                                          potatoes              0.822953
                                        vegetables              0.820009
                                              meat              0.805175
                                             dairy              0.788018
                                        sauerkraut              0.775371
                                            corned              0.759614
                                           poultry              0.756916
                                           lentils              0.742032
                                              stew              0.740848
                                               soy              0.737750
                                            mutton              0.736008
                                         marinated              0.734211
                                       cauliflower              0.731798
                                            onions              0.730454
                                            mashed              0.728839
                                           seafood              0.725203
                                             meats              0.719754
                                             offal              0.716346
                                             fried              0.715697
                                             beets              0.714562
                                          sausages              0.713819
                                           grilled              0.709343
                                           oilseed              0.708957
                                              rice              0.708570
                                             diced              0.707306
                                         vegetable              0.706809
                                          soybeans              0.700331
                                            sliced              0.699291
                                            shrimp              0.697173
                                            cheese              0.695822
                                            cooked              0.693799
                                          tomatoes              0.693536
                                             sauce              0.688957
                                             beans              0.684833
                                          zucchini              0.683065
                                            stewed              0.681175
                                            salami              0.679339
                                            manioc              0.678979
                                              tofu              0.678386
Enter word or sentence (EXIT to break):
```

准确率



```
[root@hcss-ecs-4b66 trunk]# ./demo-word-accuracy.sh
make: Nothing to be done for 'all'.
capital-common-countries:
ACCURACY TOP1: 85.18 %  (431 / 506)
Total accuracy: 85.18 %   Semantic accuracy: 85.18 %   Syntactic accuracy: -nan %
capital-world:
ACCURACY TOP1: 74.10 %  (1076 / 1452)
Total accuracy: 76.97 %   Semantic accuracy: 76.97 %   Syntactic accuracy: -nan %
currency:
ACCURACY TOP1: 16.42 %  (44 / 268)
Total accuracy: 69.68 %   Semantic accuracy: 69.68 %   Syntactic accuracy: -nan %
city-in-state:
ACCURACY TOP1: 47.04 %  (739 / 1571)
Total accuracy: 60.31 %   Semantic accuracy: 60.31 %   Syntactic accuracy: -nan %
family:
ACCURACY TOP1: 57.52 %  (176 / 306)
Total accuracy: 60.10 %   Semantic accuracy: 60.10 %   Syntactic accuracy: -nan %
gram1-adjective-to-adverb:
ACCURACY TOP1: 15.21 %  (115 / 756)
Total accuracy: 53.12 %   Semantic accuracy: 60.10 %   Syntactic accuracy: 15.21 %
gram2-opposite:
ACCURACY TOP1: 18.95 %  (58 / 306)
Total accuracy: 51.09 %   Semantic accuracy: 60.10 %   Syntactic accuracy: 16.29 %
gram3-comparative:
ACCURACY TOP1: 35.32 %  (445 / 1260)
Total accuracy: 48.00 %   Semantic accuracy: 60.10 %   Syntactic accuracy: 26.61 %
gram4-superlative:
ACCURACY TOP1: 18.18 %  (92 / 506)
Total accuracy: 45.82 %   Semantic accuracy: 60.10 %   Syntactic accuracy: 25.11 %
gram5-present-participle:
ACCURACY TOP1: 27.22 %  (270 / 992)
Total accuracy: 43.49 %   Semantic accuracy: 60.10 %   Syntactic accuracy: 25.65 %
gram6-nationality-adjective:
ACCURACY TOP1: 81.18 %  (1113 / 1371)
Total accuracy: 49.05 %   Semantic accuracy: 60.10 %   Syntactic accuracy: 40.32 %
gram7-past-tense:
ACCURACY TOP1: 31.38 %  (418 / 1332)
Total accuracy: 46.84 %   Semantic accuracy: 60.10 %   Syntactic accuracy: 38.49 %
gram8-plural:
ACCURACY TOP1: 64.72 %  (642 / 992)
Total accuracy: 48.36 %   Semantic accuracy: 60.10 %   Syntactic accuracy: 41.96 %
gram9-plural-verbs:
ACCURACY TOP1: 25.08 %  (163 / 650)
Total accuracy: 47.13 %   Semantic accuracy: 60.10 %   Syntactic accuracy: 40.61 %
Questions seen / total: 12268 19544   62.77 %
[root@hcss-ecs-4b66 trunk]#
```

比较与总结

配置	用时	TOTAL(ACCU)	SEMATIC(语义)	SYNTATIC(句法)
CBOW+negative	45min	53.50	58.54	50.96
CBOW+hs	15min	35.29	36.56	34.65
SG+negative	2h	51.83	65.66	44.87
SG+hs	1h	47.13	60.10	40.61

1. 负采样的准确率明显比继承Softmax要高，可能是因为采用了更多的训练样本
2. 在负采样下，CBOW的句法准确率比SG高，但语义准确率比SG低；
3. 在hs下，SG的语义与句法准确率均较高；
   这可能因为SG比CBOW有更多的输出，更多的反向传播过程。
4. 实验中，SG的用时明显CBOW要长，因为训练样本更多；
5. negative sampled的单个训练用时理应比hs要短，因为只要做一次内积，

   但实验中负采样规模较大(negative=25)，使得用时要长于 hs(假设训练时远程服务器主机的工作效率一致，因为只开了这一个进程和一个tomcat网站)

代码阅读

- demo-word.sh

全局变量

1. MAX_STRING: 单词的最大长度。
2. vocab_size: 词汇表的大小。
3. code: 二进制编码数组，用于存储词汇表中每个单词的哈夫曼编码。
4. point: 哈夫曼树中每个单词的路径，用于快速查询。
5. b_max_size: 未提供足够信息，无法确定其功能。
6. layer1_size: 神经网络隐藏层的大小。
7. train_words: 训练文本中的单词总数。
8. word_count_actual: 实际处理的单词数量。
9. iter: 迭代次数。
10. file_size: 文件大小。
11. classes: 未提供足够信息，无法确定其功能。

12. alpha、starting_alpha为学习率，sample为采样
13. syn0和neu1e为隐藏层，syn1=输出层，syn1neg为负采样的输出层，expTable为softmax函数表。

函数功能

1. InitUnigramTable()：构建一个根据词频分布进行负采样的表，以便在训练过程中高效地选择负样本。
2. ReadWord()：从文件中读取一个单词，放在word中国，如果a超过了最大字符数限制MAX_STRING-1，就将a减1，以截断过长的单词。
3. GetWordHash()：返回单词的Hash值
4. SearchVocab()：返回一个单词在词汇表中的作用
5. ReadWordIndex()：读一个单词，并返回在词汇表中的索引
6. AddWordToVocab()：往词汇表数组添加一个单词，内存越界重新分配内存；计算单词的hash值，线性探测法存储在vocab_hash中
7. VocabCompare()：比较a和b的频度
8. SortVocab()：按词频排序词汇表
9. ReduceVocab()：移除不常见的编码来降低词汇
10. CreateBinaryTree()：通过词频来构建二叉Huffman树
11. LearnVocabFromTrainFile()：从训练文件来学习词汇表
12. SaveVocab()：保存词汇表到save_vocab_file文件
13. ReadVocab()：阅读词汇表
14. InitNet()：初始化网络
15. *TrainModelThread()：

    读取当前单词序号，分别在CBOW框架中或Skip-gram框架中，使用

    hs或negative sample来更新网络
16. TrainModel()：多线程训练网络
17. main()：设置参数、训练

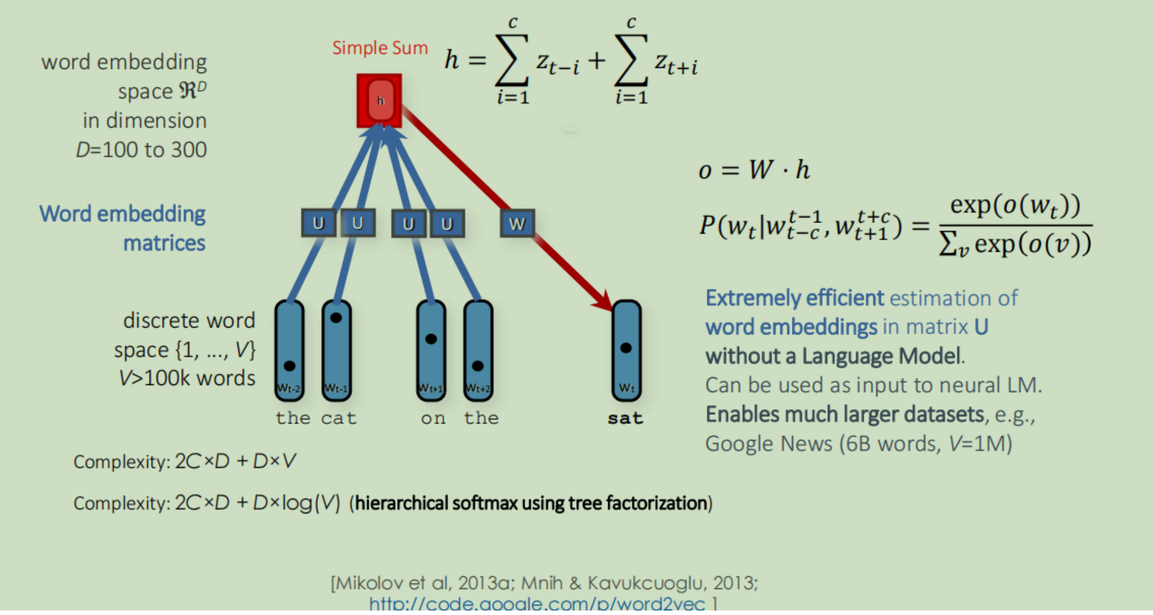
*TrainModelThread()阅读

CBOW

- 获取单词的嵌入式表示，并加入到neu1中，

```
    if (cbow) {  //train the cbow architecture
      // in -> hidden
      cw = 0;
      for (a = b; a < window * 2 + 1 - b; a++) if (a != window) {
        c = sentence_position - window + a;
        if (c < 0) continue;
        if (c >= sentence_length) continue;
        last_word = sen[c];
        if (last_word == -1) continue;
        for (c = 0; c < layer1_size; c++) neu1[c] += syn0[c + last_word * layer1_size];
        cw++;
      }
```

- 正向传播，反向更新，按如下公式：



word embedding space $\Re^D$ in dimension D=100 to 300
Word embedding matrices
discrete word space {1, ..., V} V>100k words
Simple Sum
h
U
U
U
U
W
W_{t-2}
W_{t-1}
W_{t+1}
W_{t+2}
W_t
the cat
on the
sat

$$h = \sum_{i=1}^{c} z_{t-i} + \sum_{i=1}^{c} z_{t+i}$$

$$o = W \cdot h$$

$$P(w_t | w_{t-c}^{t-1}, w_{t+1}^{t+c}) = \frac{\exp(o(w_t))}{\sum_v \exp(o(v))}$$

Extremely efficient estimation of **word embeddings** in matrix **U** **without a Language Model**. Can be used as input to neural LM. **Enables much larger datasets**, e.g., Google News (6B words, V=1M)

Complexity: $2C \times D + D \times V$

Complexity: $2C \times D + D \times \log(V)$ (**hierarchical softmax using tree factorization**)

[Mikolov et al, 2013a; Mnih & Kavukcuoglu, 2013; http://code.google.com/p/word2vec]

```
      if (cw) {
        for (c = 0; c < layer1_size; c++) neu1[c] /= cw;
        if (hs) for (d = 0; d < vocab[word].codelen; d++) {
          f = 0;
          l2 = vocab[word].point[d] * layer1_size;
          // Propagate hidden -> output
          for (c = 0; c < layer1_size; c++) f += neu1[c] * syn1[c + l2];
          if (f <= -MAX_EXP) continue;
          else if (f >= MAX_EXP) continue;
          else f = expTable[(int)((f + MAX_EXP) * (EXP_TABLE_SIZE / MAX_EXP / 2))];
          // 'g' is the gradient multiplied by the learning rate
          g = (1 - vocab[word].code[d] - f) * alpha;
          // Propagate errors output -> hidden
          for (c = 0; c < layer1_size; c++) neu1e[c] += g * syn1[c + l2];
          // Learn weights hidden -> output
          for (c = 0; c < layer1_size; c++) syn1[c + l2] += g * neu1[c];
        }
```

负采样

```
        // NEGATIVE SAMPLING
        if (negative > 0) for (d = 0; d < negative + 1; d++) {
          if (d == 0) {
            target = word;
            label = 1;
          } else {
            next_random = next_random * (unsigned long long)25214903917 + 11;
            target = table[(next_random >> 16) % table_size];
            if (target == 0) target = next_random % (vocab_size - 1) + 1;
            if (target == word) continue;
            label = 0;
          }
          l2 = target * layer1_size;
```

- 其更新公式：

```
     l2 = target * layer1_size;
     f = 0;
     for (c = 0; c < layer1_size; c++) f += neu1[c] * syn1neg[c + l2];
     if (f > MAX_EXP) g = (label - 1) * alpha;
     else if (f < -MAX_EXP) g = (label - 0) * alpha;
     else g = (label - expTable[(int)((f + MAX_EXP) * (EXP_TABLE_SIZE / MAX_EXP / 2))]) * alpha;
     for (c = 0; c < layer1_size; c++) neu1e[c] += g * syn1neg[c + l2];
     for (c = 0; c < layer1_size; c++) syn1neg[c + l2] += g * neu1[c];
    }
```

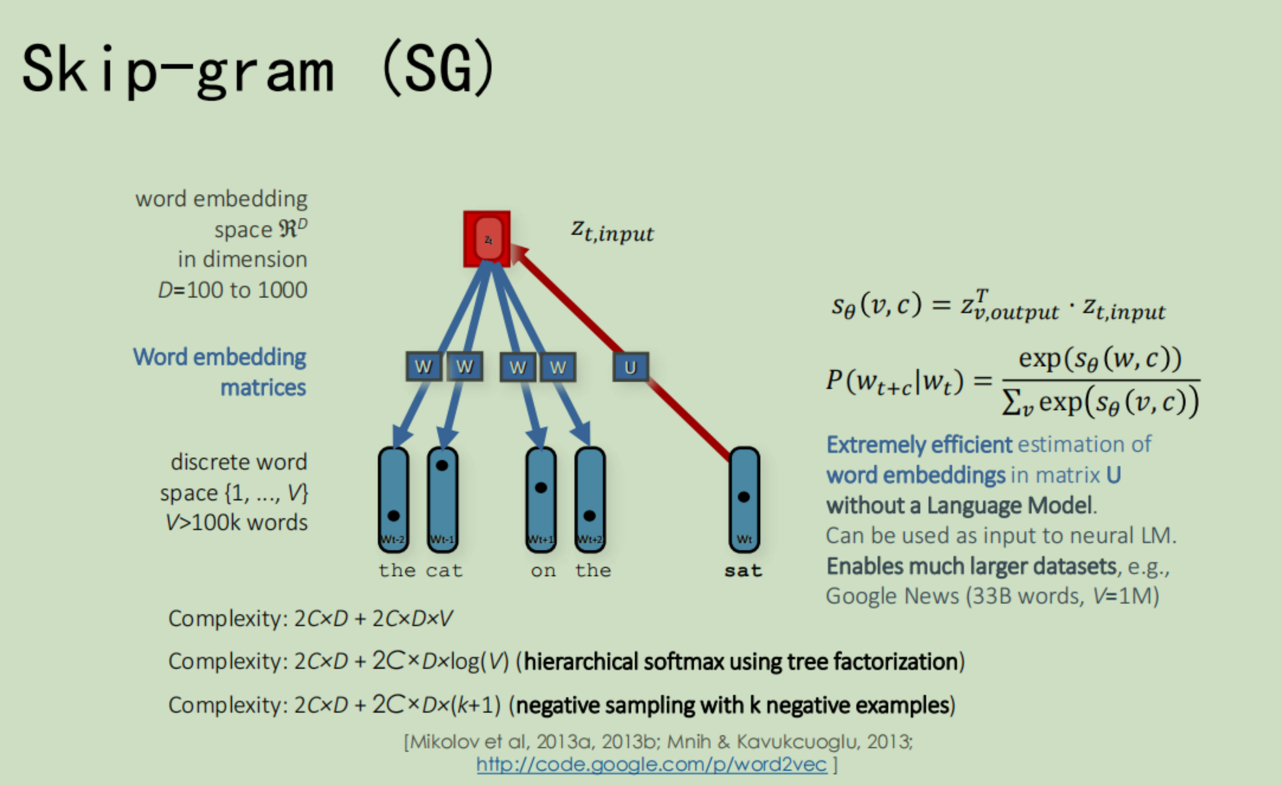
3. 隐藏层到输入层的更新

```
    // hidden -> in
    for (a = b; a < window * 2 + 1 - b; a++) if (a != window) {
     c = sentence_position - window + a;
     if (c < 0) continue;
     if (c >= sentence_length) continue;
     last_word = sen[c];
     if (last_word == -1) continue;
     for (c = 0; c < layer1_size; c++) syn0[c + last_word * layer1_size] += neu1e[c];
    }
```

SG

- 对[b,window*2-b]中不等于window的位置，找到syn0对应的输入的首地址，和syn1作内积

```
   for (a = b; a < window * 2 + 1 - b; a++) if (a != window) {
    c = sentence_position - window + a;
    if (c < 0) continue;
    if (c >= sentence_length) continue;
    last_word = sen[c];
    if (last_word == -1) continue;
    l1 = last_word * layer1_size;
```



Skip-gram (SG)

word embedding space $\Re^D$ in dimension D=100 to 1000

Word embedding matrices

discrete word space {1, ..., V} V>100k words

$$s_\theta(v,c) = z_{v,output}^T \cdot z_{t,input}$$

$$P(w_{t+c}|w_t) = \frac{\exp(s_\theta(w,c))}{\sum_v \exp(s_\theta(v,c))}$$

Extremely efficient estimation of **word embeddings** in matrix **U** **without a Language Model**.
Can be used as input to neural LM.
Enables much larger datasets, e.g., Google News (33B words, V=1M)

Complexity: $2C \times D + 2C \times D \times V$

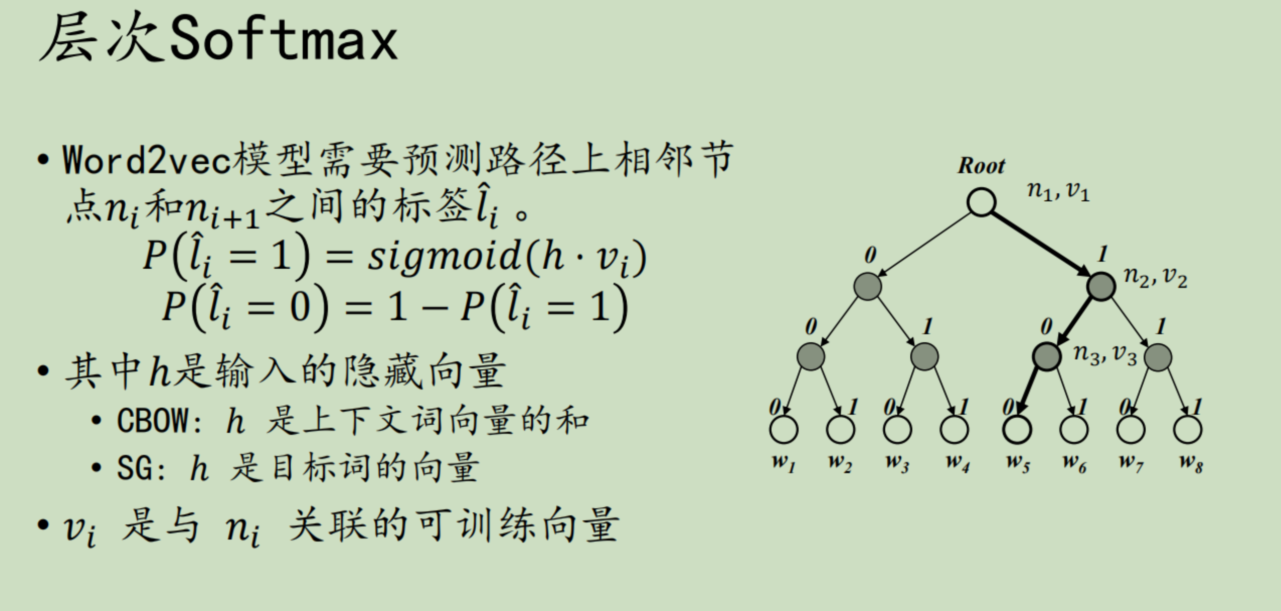
Complexity: $2C \times D + 2C \times D \times \log(V)$ (**hierarchical softmax using tree factorization**)

Complexity: $2C \times D + 2C \times D \times (k+1)$ (**negative sampling with k negative examples**)

[Mikolov et al, 2013a, 2013b; Mnih & Kavukcuoglu, 2013; http://code.google.com/p/word2vec]

层次Softmax

- 按层次Softmax，判别当前输入与输出的内积，



层次Softmax

- Word2vec模型需要预测路径上相邻节点n_i和n_{i+1}之间的标签$\hat{l}_i$。
  $$P(\hat{l}_i = 1) = sigmoid(h \cdot v_i)$$
  $$P(\hat{l}_i = 0) = 1 - P(\hat{l}_i = 1)$$
- 其中h是输入的隐藏向量
  - CBOW：h 是上下文词向量的和
  - SG：h 是目标词的向量
- v_i 是与 n_i 关联的可训练向量

- 如果在[-MAX_EXP,MAX_EXP]内，更新nn、输入、输出层；否则继续

```
        // HIERARCHICAL SOFTMAX
        if (hs) for (d = 0; d < vocab[word].codelen; d++) {
          f = 0;
          l2 = vocab[word].point[d] * layer1_size;
          // Propagate hidden -> output
          for (c = 0; c < layer1_size; c++) f += syn0[c + l1] * syn1[c + l2];
          if (f <= -MAX_EXP) continue;
          else if (f >= MAX_EXP) continue;
          else f = expTable[(int)((f + MAX_EXP) * (EXP_TABLE_SIZE / MAX_EXP / 2))];
          // 'g' is the gradient multiplied by the learning rate
          g = (1 - vocab[word].code[d] - f) * alpha;
          // Propagate errors output -> hidden
          for (c = 0; c < layer1_size; c++) neu1e[c] += g * syn1[c + l2];
          // Learn weights hidden -> output
          for (c = 0; c < layer1_size; c++) syn1[c + l2] += g * syn0[c + l1];
        }
```