Section 2:
Least Similar Pairs
(0.23, 'new', 'ancient')
(-0.041323334, 'house', 'key')

Most Similar Pairs
(9.8, 'vanish', 'disappear')
(0.9674536, 'south', 'north')

Both do not match

How do those correlation value compare to each other? [4 points]
From the correlation values, it seems like the higher the dimension, the better the correlation is with human judgement values.

Section 3.2:

Target	k	Paired F-Score
paper.n	7	0.4803
suspend.v	6	0.2830
miss.v	8	0.2098
expect.v	6	0.4205
interest.n	5	0.2960
receive.v	13	0.2289
write.v	9	0.3213
mean.v	6	0.4361
plan.n	3	0.6118
shelter.n	5	0.3914
begin.v	8	0.3431
judgment.n	7	0.3068
treat.v	8	0.3322
watch.v	5	0.4400
simple.a	5	0.2564
bank.n	9	0.2429
note.v	3	0.6400
rule.v	7	0.2911
organization.n	7	0.4051
different.a	1	1.0000
express.v	7	0.3849
source.n	9	0.2741
provide.v	7	0.5827
talk.v	6	0.5069

operate.v	7	0.2792
smell.v	4	0.5000
difference.n	5	0.4758
party.n	5	0.3271
eat.v	6	0.4084
hear.v	5	0.3234
performance.n	5	0.4402
climb.v	6	0.3275
use.v	6	0.6499
win.v	4	0.4892
image.n	9	0.2692
degree.n	7	0.4098
play.v	34	0.1783
produce.v	7	0.4310
atmosphere.n	6	0.4174
wash.v	13	0.1509

=> Average Paired F-Score: 0.3399

This method uses Agglomerative Clustering along with sparse vector representation on 500 most frequent words and a wind context of 3. It should be finding words that based on their relationships on how often they appear around each other. Some params I used are euclidian distance and single linkage to minimize distances between clusters. At first I used cosine metric but it did not give a great f-score compared to euclidean which yielded an extra 0.04. It is quite low..

Section 3.3

Target	k	Paired F-Score
paper.n	7	0.5028
suspend.v	6	0.5091
miss.v	8	0.2400
expect.v	6	0.3730
interest.n	5	0.4585
receive.v	13	0.2020
write.v	9	0.3171
mean.v	6	0.3768
plan.n	3	0.6387
shelter.n	5	0.4574
begin.v	8	0.3200
judgment.n	7	0.2498
treat.v	8	0.3200

```
|     watch.v      | 5  |      0.4638      |
|     simple.a     | 5  |      0.2857      |
|      bank.n      | 9  |      0.5000      |
|      note.v      | 3  |      0.5714      |
|      rule.v      | 7  |      0.3615      |
|  organization.n  | 7  |      0.3614      |
|   different.a    | 1  |      1.0000      |
|    express.v     | 7  |      0.4058      |
|     source.n     | 9  |      0.2712      |
|    provide.v     | 7  |      0.6246      |
|      talk.v      | 6  |      0.6114      |
|    operate.v     | 7  |      0.2848      |
|     smell.v      | 4  |      0.4348      |
|   difference.n   | 5  |      0.4776      |
|     party.n      | 5  |      0.3479      |
|      eat.v       | 6  |      0.4213      |
|      hear.v      | 5  |      0.3766      |
|  performance.n   | 5  |      0.4427      |
|     climb.v      | 6  |      0.2840      |
|      use.v       | 6  |      0.4336      |
|      win.v       | 4  |      0.3904      |
|     image.n      | 9  |      0.3082      |
|     degree.n     | 7  |      0.4283      |
|      play.v      | 34 |      0.1375      |
|    produce.v     | 7  |      0.4450      |
|   atmosphere.n   | 6  |      0.3391      |
|      wash.v      | 13 |      0.2308      |
+------------------+----+------------------+
=> Average Paired F-Score:  0.3501
```

This method uses a dense vector representation with 300 dimensions. I also used k-means for clustering. I also tried agglomerative clustering and both yielded about the same results in terms of f-scores. I wonder why both these f-scores are so low. Is it the vectors we are using or the clustering techniques and preprocessing that needs work.

Section 3.3.3

The dense and sparse vectors yield really similar and really low results which is quite odd.. I might be doing something wrong. The first clusters of each vector seem to always have the most words and also the most similar words. The following clusters after that tend to have different words in their respective clusters.

Section 3.4

Target	k	Paired F-Score
suspend.v	6	0.4878
miss.v	8	0.3058
degree.n	7	0.4276
simple.a	5	0.3902
use.v	6	0.6243
plan.n	3	0.6376
performance.n	5	0.4498
begin.v	8	0.3382
eat.v	6	0.4283
atmosphere.n	6	0.3773
interest.n	5	0.3277
rule.v	7	0.3129
climb.v	6	0.2192
organization.n	7	0.3703
source.n	9	0.2863
treat.v	8	0.3502
expect.v	6	0.4854
different.a	1	1.0000
produce.v	7	0.4494
play.v	34	0.1760
win.v	4	0.5445
hear.v	5	0.3544
judgment.n	7	0.2625
party.n	5	0.3487
shelter.n	5	0.4565
bank.n	9	0.4545
express.v	7	0.4084
mean.v	6	0.3955
receive.v	13	0.2347
operate.v	7	0.3060
paper.n	7	0.5586
watch.v	5	0.4567
image.n	9	0.3080
difference.n	5	0.4774
talk.v	6	0.6143
write.v	9	0.3208
note.v	3	0.2941
smell.v	4	0.5045
provide.v	7	0.6361

```
|     wash.v     | 13 |     0.2349     |
+---------------+----+---------------+
=> Average Paired F-Score:  0.3657
```

I chose the 300 dimensional dense vector from google. I then generate random vectors to match the dimensions for the clustering process. I used K means clustering where I utilized silhouette scores to find the best number of clusters k. A good silhouette score means a quality cluster. The F-scores are again quite low at 0.36. I wish I had a little more time to spend on this hw but I will revisit it and try more techniques later.