AMS 691.02: Natural Language Processing – Fall 2024
Assignment 1: Distributional Word Vectors

REPORT

1)Distributional Counting

1.1) I have written a code to count the number of times a word y appears in a context window with size w centered at the word x, using the provided **wiki-1percent.txt** corpus. A context window contains up to w words to either side of the center word, so it contains up to 2w + 1 words in total (including the center word).

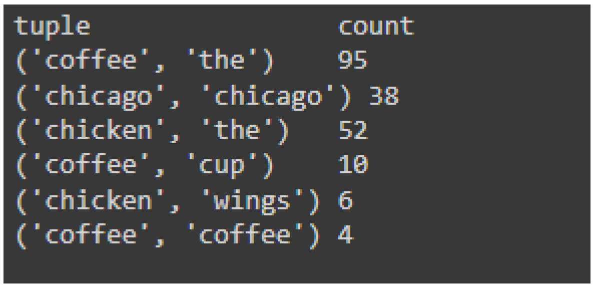
I have counted each occurrence of y in a single window and used the notation #(x, y) to denote the count of the tuple ⟨x, y⟩, i.e., the number of times that word y appeared within w words to the left or right of x. Here tuples are ordered: the first item in the tuple ⟨x, y⟩ is the center word and the second item is the context word.

Implementation:

Iterated through all valid windows in each line (a valid window is one that is centered on a token in a line). For each valid window centered on word x, iterate through the w words to the left of x and the w words to the right of x. Done this by finding the index of each word x and according to the window size w calculated ind_l and ind_L for the left window and similarly ind_r and ind_R for the right window. Then, for each such word y, incremented the count for the tuple ⟨x, y⟩. Used V (for vocabulary) and VC (context vocabulary) when computing these counts.

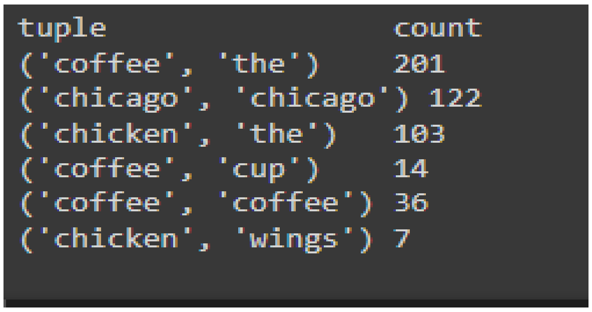
1.2) Using vocab-15kws.txt to populate V and vocab-5k.txt to populate VC, use your code to report #(x, y) for the pairs in the following table for both w = 3 and w = 6. Some counts have already been filled in for you which you can use to check your code:

For w = 3:



```
tuple                    count
('coffee', 'the')        95
('chicago', 'chicago') 38
('chicken', 'the')       52
('coffee', 'cup')        10
('chicken', 'wings') 6
('coffee', 'coffee') 4
```

For w = 6:

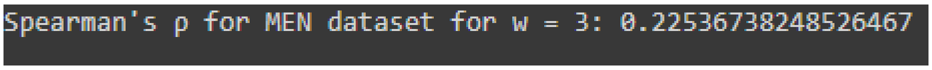


```
tuple                    count
('coffee', 'the')        201
('chicago', 'chicago') 122
('chicken', 'the')       103
('coffee', 'cup')        14
('coffee', 'coffee') 36
('chicken', 'wings') 7
```

1.3) Using w = 3 (and again using vocab-15kws.txt for V and vocab-5k.txt for VC), evaluate your count-based word vectors using EVALWS and report your results on MEN and SimLex-999.

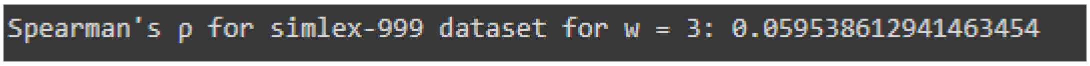
For w = 3:

For MEN dataset:



```
Spearman's ρ for MEN dataset for w = 3: 0.22536738248526467
```

For the simlex-999 dataset:



```
Spearman's ρ for simlex-999 dataset for w = 3: 0.059538612941463454
```

Here for both the datasets, the correlation is low but still, the MEN dataset has less noise and more informative words than the simlex-999 dataset. That is why Spearman's correlation is almost 0 for the second dataset.

2) Combining Counts with Inverse Document Frequency (IDF)

2.1) Extend your implementation to be able to compute IDF-based word vectors using Eq. 1. Using w = 3, vocab-15kws.txt to populate V, and vocab-5k.txt to populate VC, evaluate (EVALWS) your IDF-based word vectors and report your results.

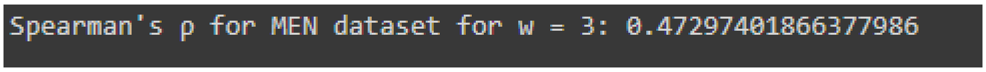
Here I have used sentence retrieval, rather than documents as we are working with sentences. Let S denote the set of sentences in the corpus. Then, instead of defining word vector entries using counts #(x, y), I have defined them as follows. The word vector for a word x ∈ V has an entry for each word y ∈ VC with a value given by:

#(x, y) × (|S| / |{s ∈ S: s contains y}|)

The first term above is the "term frequency" (TF) and the second is the inverse of the "sentence frequency" for the context word.

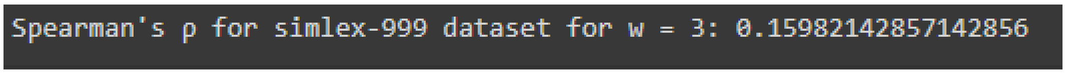
For w = 3:

For MEN dataset:



```
Spearman's ρ for MEN dataset for w = 3: 0.47297401866377986
```

For the simlex-999 dataset:

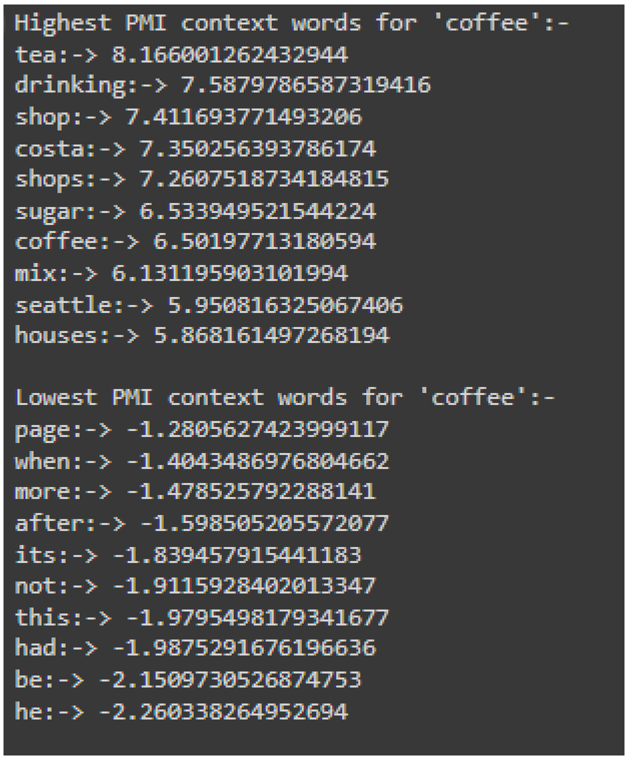


```
Spearman's ρ for simlex-999 dataset for w = 3: 0.15982142857142856
```

Here the MEN dataset shows a moderate positive correlation and it is greater than the correlation we got from the raw count method that I have used in (1.3). It is because of discounting the most frequent words like is, etc. But still, the second dataset has a low correlation. It might be because of the quality of words in the dataset or it might contain frequently used words more.

3) Pointwise Mutual Information (PMI)

3.1) Implement the capability of computing PMIs. Use your code to calculate PMIs for w = 3 when using vocab-15kws.txt to populate V and vocab-5k.txt to populate VC. Note that since we are using different vocabularies for center words and context words, pmi(a, b) will not necessarily equal pmi(b, a) (though they will be similar). (If there is a word in V that has no counts, the numerator and denominator for all of its PMI values will be zero, so you can just define all such PMIs to be zero.) For center word x = "coffee", print the 10 context words with the largest PMIs and the 10 context words with the smallest PMIs. Print both the words and the PMI values.



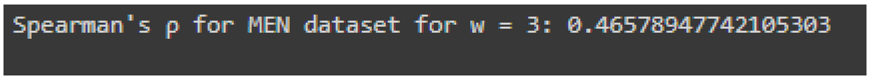
```
Highest PMI context words for 'coffee':-
tea:-> 8.166001262432944
drinking:-> 7.5879786587319416
shop:-> 7.411693771493206
costa:-> 7.350256393786174
shops:-> 7.2607518734184815
sugar:-> 6.533949521544224
coffee:-> 6.50197713180594
mix:-> 6.131195903101994
seattle:-> 5.950816325067406
houses:-> 5.868161497268194

Lowest PMI context words for 'coffee':-
page:-> -1.2805627423999117
when:-> -1.4043486976804662
more:-> -1.478525792288141
after:-> -1.598505205572077
its:-> -1.839457915441183
not:-> -1.9115928402013347
this:-> -1.9795498179341677
had:-> -1.9875291676196636
be:-> -2.1509730526874753
he:-> -2.260338264952694
```

Here I have the PMI method to calculate the count which is using joint probabilities and partial probabilities of x and y. From the snippet above, we can see the highest PMI context words and lowest PMI context words that are related to coffee using this method.

3.2) Now, define word vectors using PMI. That is, the word vector for a word x ∈ V has an entry for each word y ∈ VC with a value given by PMI (x, y). As above, use w = 3, vocab-15kws.txt to populate V, and vocab-5k.txt to populate VC. Evaluate (EVALWS) your PMI-based word vectors and report your results.

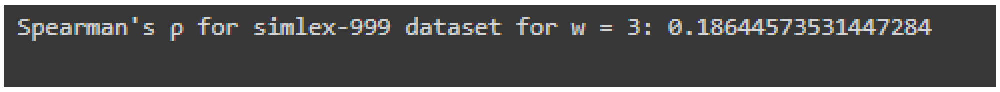
For w = 3:

For MEN dataset:



```
Spearman's ρ for MEN dataset for w = 3: 0.46578947742105303
```

For the simlex-999 dataset:



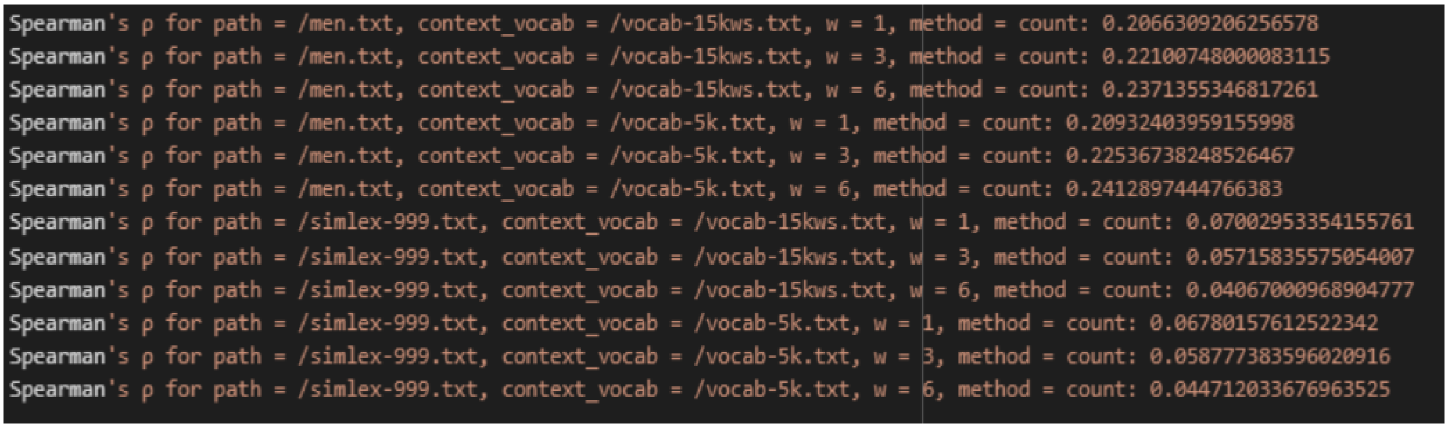
```
Spearman's ρ for simlex-999 dataset for w = 3: 0.18644573531447284
```

Here the MEN dataset shows a moderate positive correlation and it is almost equal to the correlation we got from the TF-IDF method that I have used in (2.1). There is a slight improvement in the second dataset when compared to the TF-IDF method.

4) Quantitative Comparisons

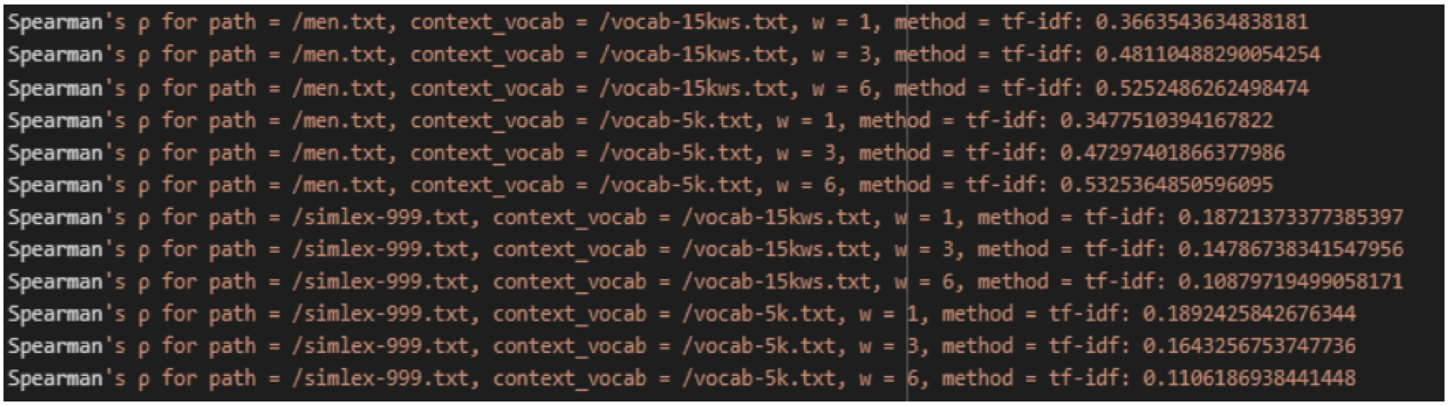
4.1) Evaluate the word vectors (EVALWS) corresponding to the three ways of computing vectors (counts, IDF, and PMI), three values of w (1, 3, and 6), and two context vocabularies (vocab-15kws.txt and vocab-5k.txt). For all cases, use vocab-15kws.txt for V. Report the results (there should be 36 correlations in all) and describe your findings.

For counts:



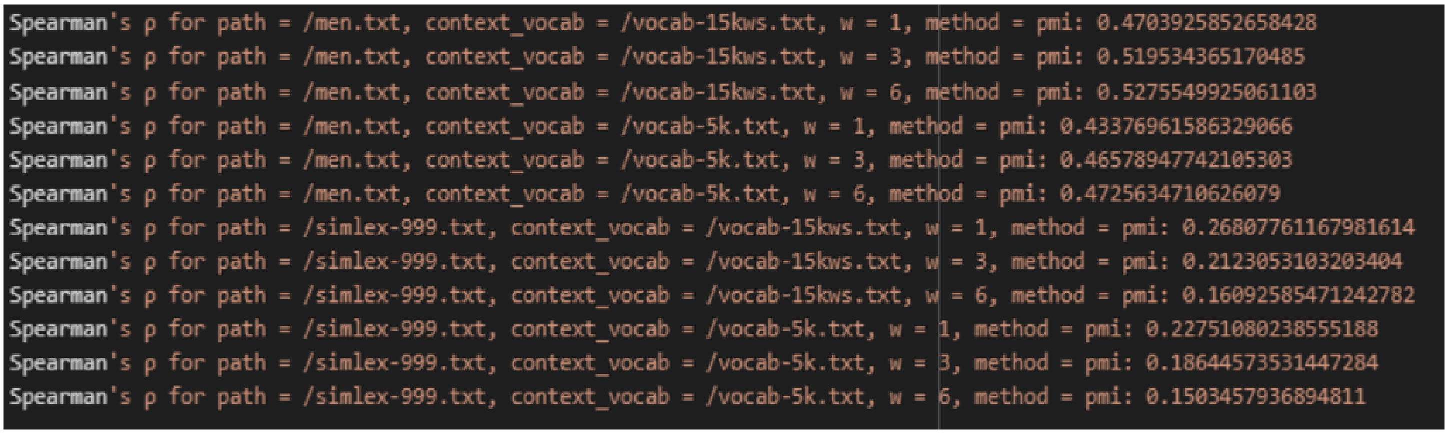
```
Spearman's ρ for path = /men.txt, context_vocab = /vocab-15kws.txt, w = 1, method = count: 0.2066309206256578
Spearman's ρ for path = /men.txt, context_vocab = /vocab-15kws.txt, w = 3, method = count: 0.22100748000083115
Spearman's ρ for path = /men.txt, context_vocab = /vocab-15kws.txt, w = 6, method = count: 0.2371355346817261
Spearman's ρ for path = /men.txt, context_vocab = /vocab-5k.txt, w = 1, method = count: 0.20932403959155998
Spearman's ρ for path = /men.txt, context_vocab = /vocab-5k.txt, w = 3, method = count: 0.22536738248526467
Spearman's ρ for path = /men.txt, context_vocab = /vocab-5k.txt, w = 6, method = count: 0.2412897444766383
Spearman's ρ for path = /simlex-999.txt, context_vocab = /vocab-15kws.txt, w = 1, method = count: 0.07002953354155761
Spearman's ρ for path = /simlex-999.txt, context_vocab = /vocab-15kws.txt, w = 3, method = count: 0.05715835575054007
Spearman's ρ for path = /simlex-999.txt, context_vocab = /vocab-15kws.txt, w = 6, method = count: 0.04067000968904777
Spearman's ρ for path = /simlex-999.txt, context_vocab = /vocab-5k.txt, w = 1, method = count: 0.06780157612522342
Spearman's ρ for path = /simlex-999.txt, context_vocab = /vocab-5k.txt, w = 3, method = count: 0.058777383596020916
Spearman's ρ for path = /simlex-999.txt, context_vocab = /vocab-5k.txt, w = 6, method = count: 0.044712033676963525
```

For IDF:



```
Spearman's ρ for path = /men.txt, context_vocab = /vocab-15kws.txt, w = 1, method = tf-idf: 0.3663543634838181
Spearman's ρ for path = /men.txt, context_vocab = /vocab-15kws.txt, w = 3, method = tf-idf: 0.48110488290054254
Spearman's ρ for path = /men.txt, context_vocab = /vocab-15kws.txt, w = 6, method = tf-idf: 0.5252486262498474
Spearman's ρ for path = /men.txt, context_vocab = /vocab-5k.txt, w = 1, method = tf-idf: 0.3477510394167822
Spearman's ρ for path = /men.txt, context_vocab = /vocab-5k.txt, w = 3, method = tf-idf: 0.47297401866377986
Spearman's ρ for path = /men.txt, context_vocab = /vocab-5k.txt, w = 6, method = tf-idf: 0.5325364850596095
Spearman's ρ for path = /simlex-999.txt, context_vocab = /vocab-15kws.txt, w = 1, method = tf-idf: 0.18721373377385397
Spearman's ρ for path = /simlex-999.txt, context_vocab = /vocab-15kws.txt, w = 3, method = tf-idf: 0.14786738341547956
Spearman's ρ for path = /simlex-999.txt, context_vocab = /vocab-15kws.txt, w = 6, method = tf-idf: 0.10879719499058171
Spearman's ρ for path = /simlex-999.txt, context_vocab = /vocab-5k.txt, w = 1, method = tf-idf: 0.1892425842676344
Spearman's ρ for path = /simlex-999.txt, context_vocab = /vocab-5k.txt, w = 3, method = tf-idf: 0.1643256753747736
Spearman's ρ for path = /simlex-999.txt, context_vocab = /vocab-5k.txt, w = 6, method = tf-idf: 0.1106186938441448
```

For PMI:



```
Spearman's ρ for path = /men.txt, context_vocab = /vocab-15kws.txt, w = 1, method = pmi: 0.4703925852658428
Spearman's ρ for path = /men.txt, context_vocab = /vocab-15kws.txt, w = 3, method = pmi: 0.519534365170485
Spearman's ρ for path = /men.txt, context_vocab = /vocab-15kws.txt, w = 6, method = pmi: 0.5275549925061103
Spearman's ρ for path = /men.txt, context_vocab = /vocab-5k.txt, w = 1, method = pmi: 0.43376961586329066
Spearman's ρ for path = /men.txt, context_vocab = /vocab-5k.txt, w = 3, method = pmi: 0.46578947742105303
Spearman's ρ for path = /men.txt, context_vocab = /vocab-5k.txt, w = 6, method = pmi: 0.4725634710626079
Spearman's ρ for path = /simlex-999.txt, context_vocab = /vocab-15kws.txt, w = 1, method = pmi: 0.26807761167981614
Spearman's ρ for path = /simlex-999.txt, context_vocab = /vocab-15kws.txt, w = 3, method = pmi: 0.2123053103203404
Spearman's ρ for path = /simlex-999.txt, context_vocab = /vocab-15kws.txt, w = 6, method = pmi: 0.16092585471242782
Spearman's ρ for path = /simlex-999.txt, context_vocab = /vocab-5k.txt, w = 1, method = pmi: 0.22751080238555188
Spearman's ρ for path = /simlex-999.txt, context_vocab = /vocab-5k.txt, w = 3, method = pmi: 0.18644573531447284
Spearman's ρ for path = /simlex-999.txt, context_vocab = /vocab-5k.txt, w = 6, method = pmi: 0.1503457936894811
```

Observations:

1) As window size increases (w = 1 to w = 6), Spearman correlations improve across all methods (count, TF-IDF, PMI) for the MEN dataset. The larger the windows better the semantic relations. The count-based method in MEN improves from 0.206 (w = 1) to 0.237 (w = 6). The TF-IDF-based method in MEN improves from 0.366 (w = 1) to 0.525 (w = 6). The PMI-based method in MEN improves from 0.470 (w = 1) to 0.527 (w = 6). But as window size increases (w = 1 to w = 6), Spearman correlations do not improve across all methods (count, TF-IDF, PMI) for the SIMLEX-999 dataset. The larger the windows worse the semantic relations. The count-based method in the SIMLEX-999 degrades from 0.07 (w = 1) to 0.04 (w = 6). The TF-IDF-based method in the SIMLEX-999 degrades from 0.187 (w = 1) to 0.108 (w = 6). The PMI-based method in the SIMLEX-999 degrades from 0.26 (w = 1) to 0.167 (w = 6). This is when I am using the vocabulary and context vocabulary the same i.e. vocab-15kws.txt. A similar trend is shown when context vocabulary is vocab-5kws.txt. So for semantic relationship MEN dataset is better than the SIMLEX-999 dataset.

2) A larger vocabulary (vocab-15k) results in higher correlations in comparison to a smaller one (vocab-5k) across all methods except the counts method which is almost the same. For ex: the PMI method with vocab-15kws.txt for MEN (w = 6) has a correlation of 0.527, compared to 0.472 with vocab-5kws.txt.
3) MEN consistently depicts higher correlations than SimLex-999 across all three methods.
4) **Count-based** method shows the lowest performance, limited by raw counts. The **TF-IDF-based** method shows better performance by reducing the impact of frequent words. The **PMI-based** method shows the best performance overall, giving weightage to valuable co-occurrences.

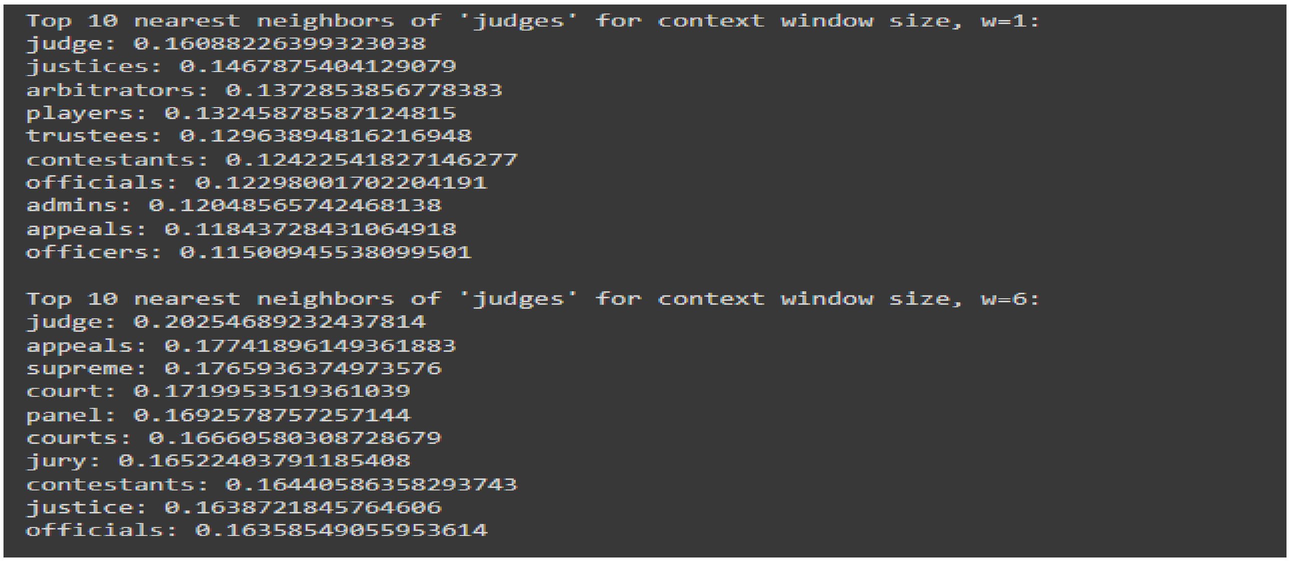
4.2) You should observe systematic trends in terms of correlation as window size changes which should differ for MEN and SimLex-999. Look at some of the manually annotated similarities in the MEN and SimLex-999 datasets and describe why you think the two datasets show the trends they do. Are these two datasets encoding the same type of similarity? How does the notion of similarity differ between them?

Observations:

1) In the **MEN** dataset as the window size increases, correlations in the MEN dataset improve across all methods, indicating it captures broader semantic contexts. This is because MEN depicts general word similarity and semantic relations. Larger window sizes provide richer co-occurrence data, enhancing the model's ability to reflect human judgments of similarity. In the **SimLex-999** dataset, shows lower correlations, even with larger window sizes. It strictly focuses on semantic similarity (e.g., "truck" and "car"), excluding functional similarity (e.g., "truck" and "road"). While larger windows improve correlation slightly, they also introduce noise by capturing functionally related but semantically unrelated words.
2) The **MEN** dataset captures a broad concept of similarity, encompassing both functional and semantic relationships. This means that words frequently co-occurring, even if not semantically related, can still be deemed similar. **The simLex-999** dataset measures strict semantic similarity, assessing words based solely on their meanings rather than functional relationships.

5) Qualitative Analysis

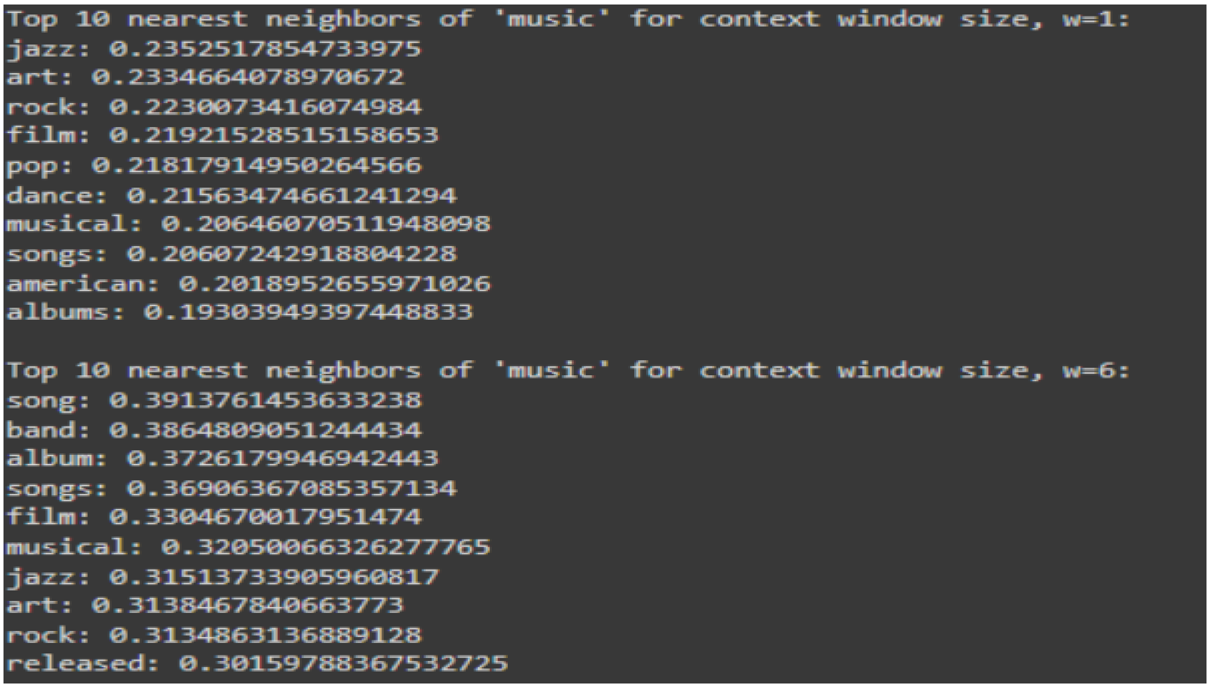
5.1) For the two window sizes w = 1 and w = 6, compute and print the 10 nearest neighbors for the query word judges. (Hint: using my implementation, the nearest neighbor for both window sizes is judge, followed by justices for w = 1 and appeals with w = 6.



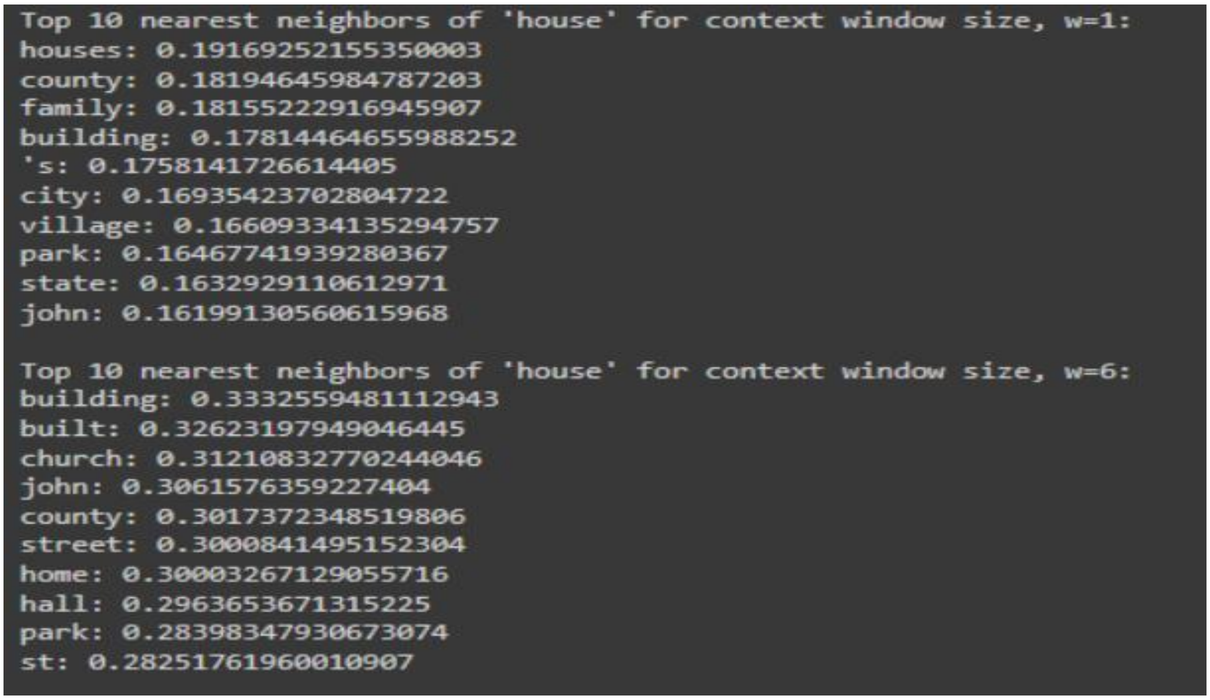
```
Top 10 nearest neighbors of 'judges' for context window size, w=1:
judge: 0.16088226399323038
justices: 0.1467875404129079
arbitrators: 0.1372853856778383
players: 0.13245878587124815
trustees: 0.12963894816216948
contestants: 0.12422541827146277
officials: 0.12298001702204191
admins: 0.12048565742468138
appeals: 0.11843728431064918
officers: 0.11500945538099501

Top 10 nearest neighbors of 'judges' for context window size, w=6:
judge: 0.20254689232437814
appeals: 0.17741896149361883
supreme: 0.1765936374973576
court: 0.1719953519361039
panel: 0.1692578757257144
courts: 0.16660580308728679
jury: 0.16522403791185408
contestants: 0.16440586358293743
justice: 0.1638721845764606
officials: 0.16358549055953614
```

5.2) Discuss your findings, showing examples of nearest neighbors for particular words to support your claims.

1) Nouns:



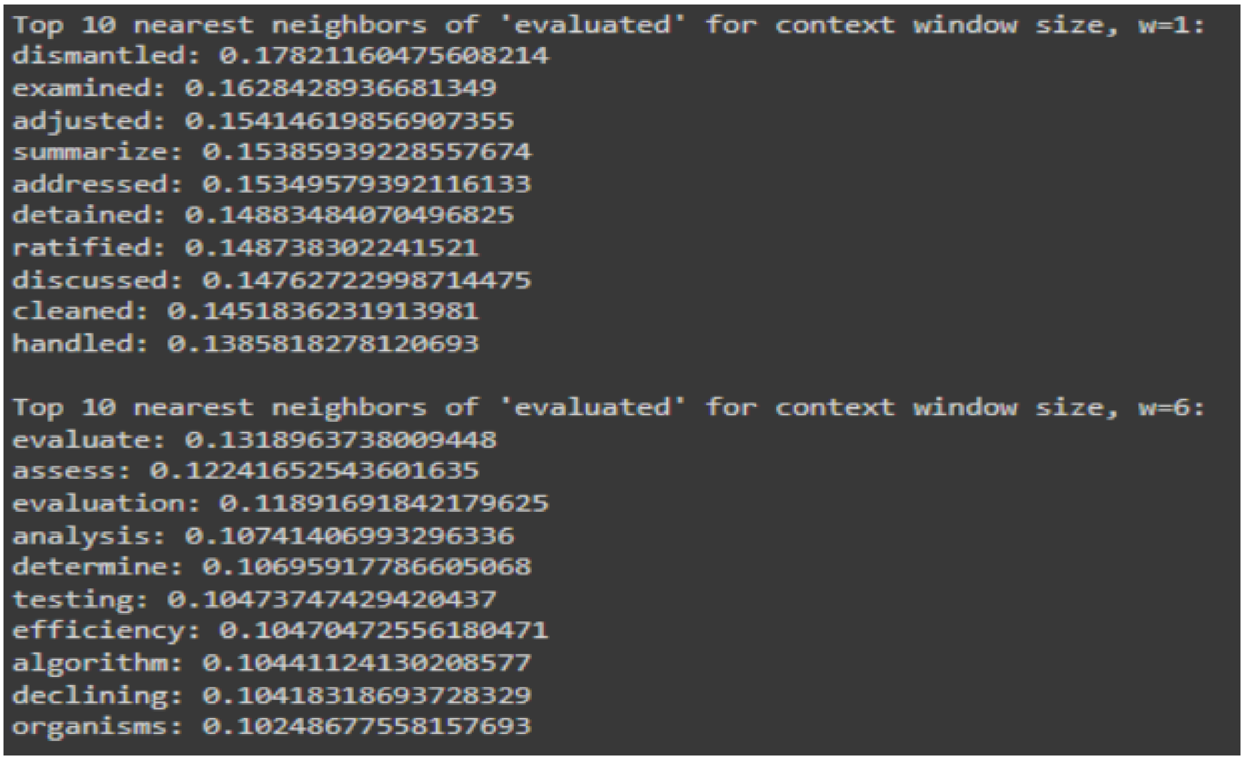
```
Top 10 nearest neighbors of 'music' for context window size, w=1:
jazz: 0.23525178547339975
art: 0.2334664078970672
rock: 0.2230073416074984
film: 0.21921528515158653
pop: 0.21817914950264566
dance: 0.21563474661241294
musical: 0.20646070511948098
songs: 0.20607242918804228
american: 0.2018952655971026
albums: 0.19303949397448833

Top 10 nearest neighbors of 'music' for context window size, w=6:
song: 0.3913761453633238
band: 0.3864809051244434
album: 0.3726179946942443
songs: 0.36906367085357134
film: 0.3304670017951474
musical: 0.32050066326277765
jazz: 0.3151373390596081 7
art: 0.3138467840663773
rock: 0.3134863136889128
released: 0.30159788367532725
```



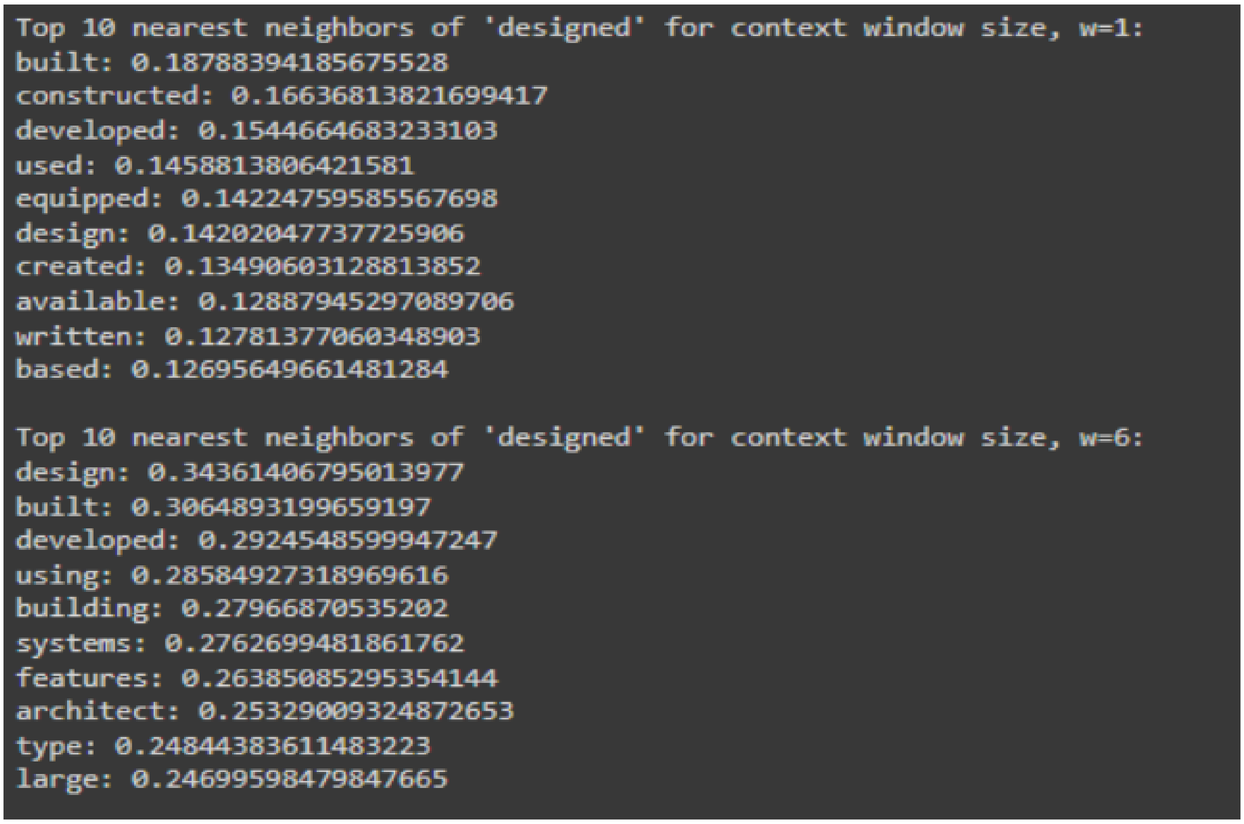
```
 Top 10 nearest neighbors of 'house' for context window size, w=1:
 houses: 0.1916925215535003
 county: 0.1819464598478720 3
 family: 0.1815522291694590 7
 building: 0.17814464655988252
 's: 0.1758141726614405
 city: 0.16935423702804722
 village: 0.16609334135294757
 park: 0.1646774193928036 7
 state: 0.1632929110612971
 john: 0.16199130560615968

 Top 10 nearest neighbors of 'house' for context window size, w=6:
 building: 0.3332559481112943
 built: 0.3262319794904644 5
 church: 0.3121083277024404 6
 john: 0.3061576359227404
 county: 0.3017372348519806
 street: 0.3000841495152304
 home: 0.3000326712905571 6
 hall: 0.296365367131522 5
 park: 0.2839834793067307 4
 st: 0.2825176196001090 7
```

For window size w = 1, nearest neighbors for "music" include nouns like jazz, art, and film, which are related to genres or forms of media. For "house", neighbors are houses, county, and family—mostly concrete nouns and place names. For Window size w = 6, the neighbors of "music" shift to song, band, and album, which are semantically more similar and specific to the domain of music. For "house", neighbors like building, church, and home become more prominent, showcasing a tighter focus on physical structures.

2) Verbs:



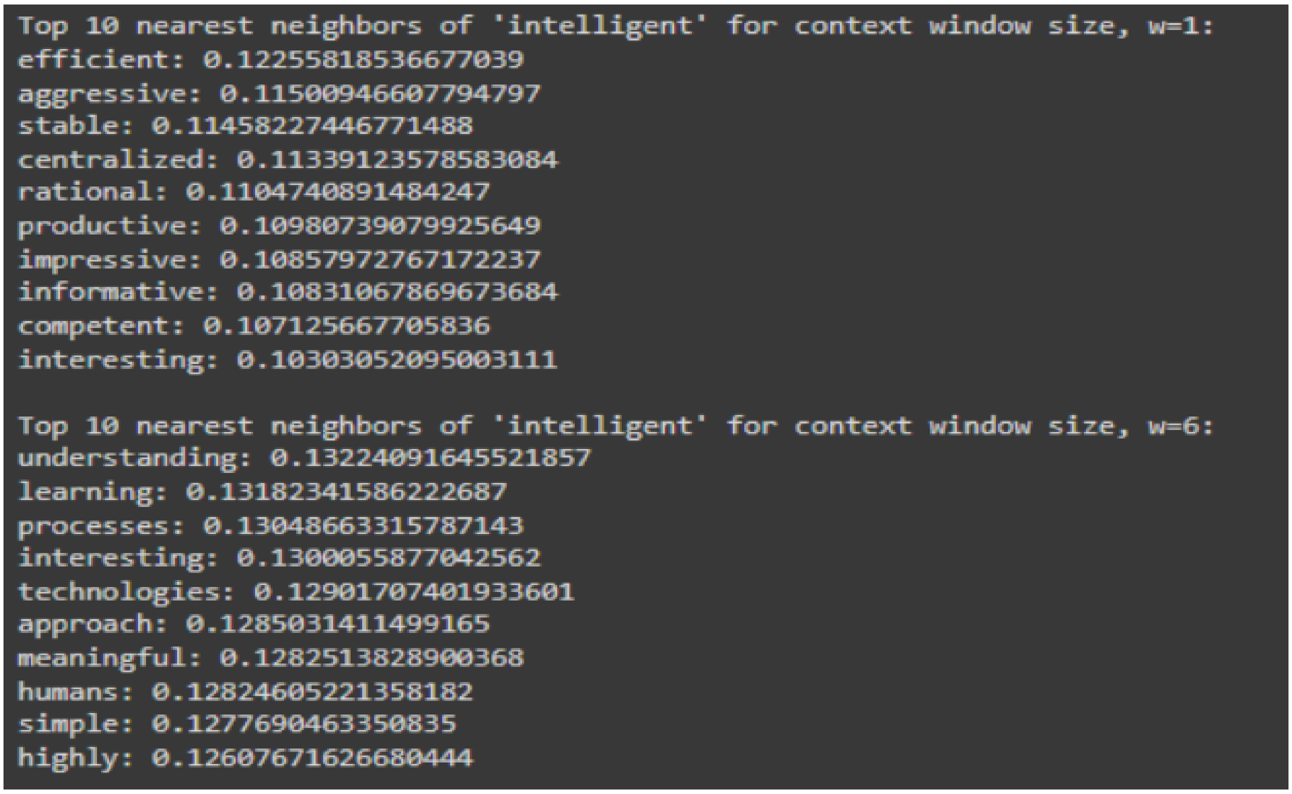
```
Top 10 nearest neighbors of 'evaluated' for context window size, w=1:
dismantled: 0.17821160475608214
examined: 0.1628428936681349
adjusted: 0.15414619856907355
summarize: 0.15385939228557674
addressed: 0.15349579392116133
detained: 0.14883484070496825
ratified: 0.148738302241521
discussed: 0.14762722998714475
cleaned: 0.1451836231913981
handled: 0.1385818278120693

Top 10 nearest neighbors of 'evaluated' for context window size, w=6:
evaluate: 0.1318963738009448
assess: 0.12241652543601635
evaluation: 0.11891691842179625
analysis: 0.10741406993296336
determine: 0.10695917786605068
testing: 0.10473747429420437
efficiency: 0.10470472556180471
algorithm: 0.10441124130208577
declining: 0.10418318693728329
organisms: 0.10248677558157693
```



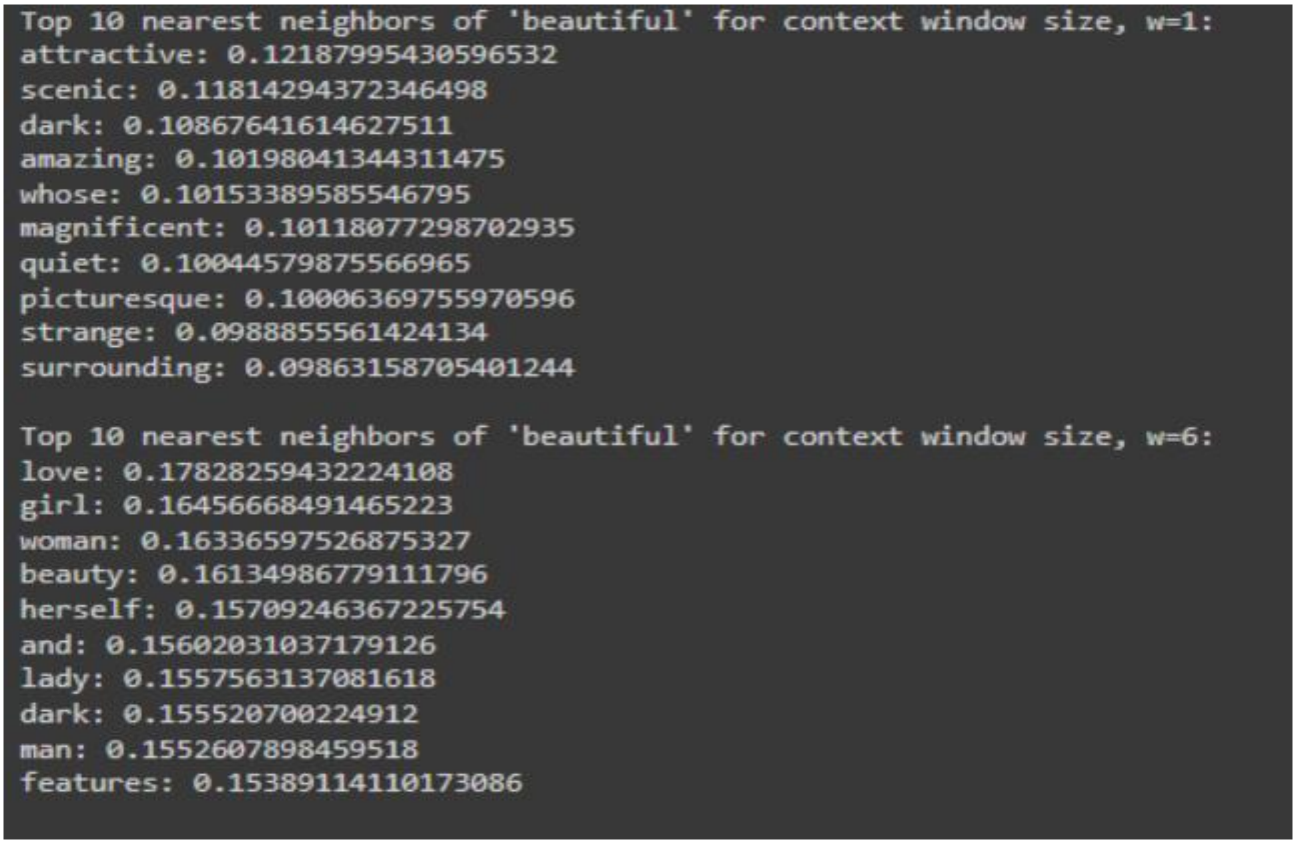
```
Top 10 nearest neighbors of 'designed' for context window size, w=1:
built: 0.18788394185675528
constructed: 0.16636813821699417
developed: 0.1544664683233103
used: 0.1458813806421581
equipped: 0.14224759585567698
design: 0.14202047737725906
created: 0.13490603128813852
available: 0.12887945297089706
written: 0.12781377060348903
based: 0.12695649661481284

Top 10 nearest neighbors of 'designed' for context window size, w=6:
design: 0.34361406795013977
built: 0.3064893199659197
developed: 0.2924548599947247
using: 0.28584927318969616
building: 0.27966870535202
systems: 0.2762699481861762
features: 0.26385085295354144
architect: 0.25329009324872653
type: 0.24844383611483223
large: 0.24699598479847665
```

For window size w = 1, the "evaluated" verb has neighbors like examined, adjusted, and dismantled, all verbs with related meanings in the context of assessment. For "designed", neighbors like built, constructed, and developed share the same part of speech(POS) (verbs). For Window size w = 6, neighbors for "evaluated" shift to evaluate, assess, and determine which are synonyms that still maintain the same POS but focus on assessment tasks. "Designed" has neighbors like design, built, and developed, still verbs but now leaning towards architectural engineering terms.

3) Adjectives:



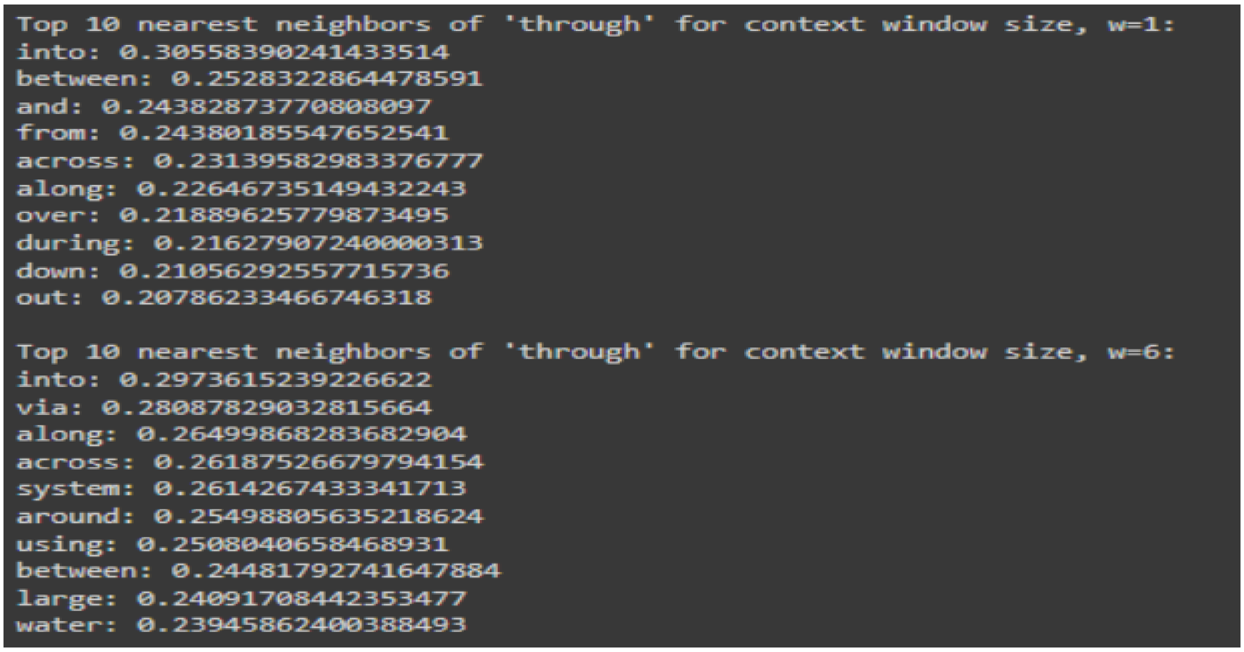
```
Top 10 nearest neighbors of 'intelligent' for context window size, w=1:
efficient: 0.12255818536677039
aggressive: 0.11500946607794797
stable: 0.11458227446771488
centralized: 0.11339123578583084
rational: 0.1104740891484247
productive: 0.10980739079925649
impressive: 0.10857972767172237
informative: 0.10831067869673684
competent: 0.107125667705836
interesting: 0.10303052095003111

Top 10 nearest neighbors of 'intelligent' for context window size, w=6:
understanding: 0.13224091645521857
learning: 0.13182341586222687
processes: 0.13048663315787143
interesting: 0.1300055877042562
technologies: 0.12901707401933601
approach: 0.1285031411499165
meaningful: 0.1282513828900368
humans: 0.12824605221358182
simple: 0.1277690463350835
highly: 0.12607671626680444
```



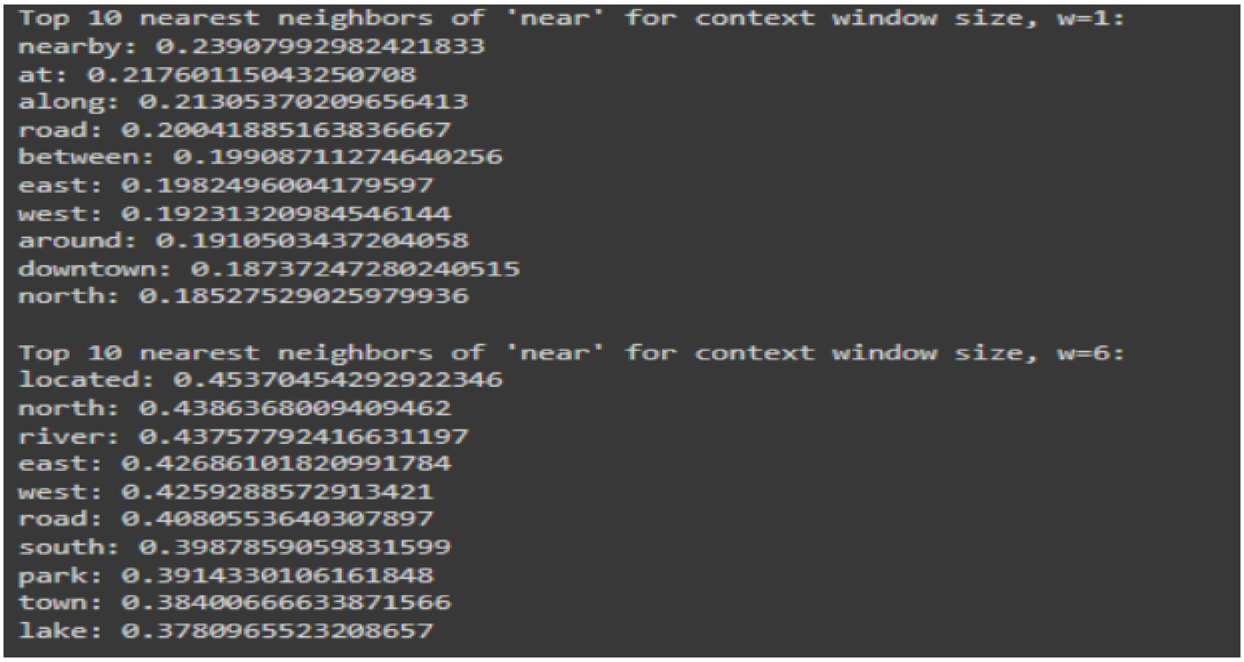
```
Top 10 nearest neighbors of 'beautiful' for context window size, w=1:
attractive: 0.12187995430596532
scenic: 0.11814294372346498
dark: 0.10867641614627511
amazing: 0.10198041344311475
whose: 0.10153389585546795
magnificent: 0.10118077298702935
quiet: 0.10044579875566965
picturesque: 0.10006369755970596
strange: 0.0988855561424134
surrounding: 0.09863158705401244

Top 10 nearest neighbors of 'beautiful' for context window size, w=6:
love: 0.17828259432224108
girl: 0.16456668491465223
woman: 0.16336597526875327
beauty: 0.16134986779111796
herself: 0.15709246367225754
and: 0.15602031037179126
lady: 0.1557563137081618
dark: 0.155520700224912
man: 0.1552607898459518
features: 0.15389114110173086
```

For window size w = 1, neighbors for "intelligent" include adjectives like efficient, stable, and productive, indicating functional or related characterstics. For "beautiful", neighbors like attractive, scenic, and amazing reflect appearance. For window size w = 6, neighbors for "intelligent" become more abstract, like understanding, learning, and processes—still conceptually related but now including more nouns. For "beautiful", neighbors like love, woman, and herself shift to more sentimental meaning.

4) Prepositions:



```
Top 10 nearest neighbors of 'through' for context window size, w=1:
into: 0.30558390241433514
between: 0.2528322864478591
and: 0.24382873770808097
from: 0.24380185547652541
across: 0.23139582983376777
along: 0.22646735149432243
over: 0.21889625779873495
during: 0.21627907240000313
down: 0.21056292557715736
out: 0.20786233466746318

Top 10 nearest neighbors of 'through' for context window size, w=6:
into: 0.2973615239226622
via: 0.28087829032815664
along: 0.26499868283682904
across: 0.26187526679794154
system: 0.2614267433341713
around: 0.25498805635218624
using: 0.2508040658468931
between: 0.24481792741647884
large: 0.24091708442353477
water: 0.23945862400388493
```



```
Top 10 nearest neighbors of 'near' for context window size, w=1:
nearby: 0.23907992982421833
at: 0.21760115043250708
along: 0.21305370209656413
road: 0.20041885163836667
between: 0.19908711274640256
east: 0.1982496004179597
west: 0.19231320984546144
around: 0.1910503437204058
downtown: 0.18737247280240515
north: 0.18527529025979936

Top 10 nearest neighbors of 'near' for context window size, w=6:
located: 0.45370454292922346
north: 0.4386368009409462
river: 0.43757792416631197
east: 0.42686101820991784
west: 0.4259288572913421
road: 0.4080553640307897
south: 0.3987859059831599
park: 0.3914330106161848
town: 0.38400666633871566
lake: 0.3780965523208657
```

For window size w = 1, neighbors for "through" include into, between, and from, all prepositions. For "near", neighbors are nearby, at, and along—also prepositions indicating location. For window size w = 6, neighbors of "through" like into, via, and along remain prepositions but with broader contextual relevance. For "near", neighbors include located, river, and town, indicating nouns related to positioning.

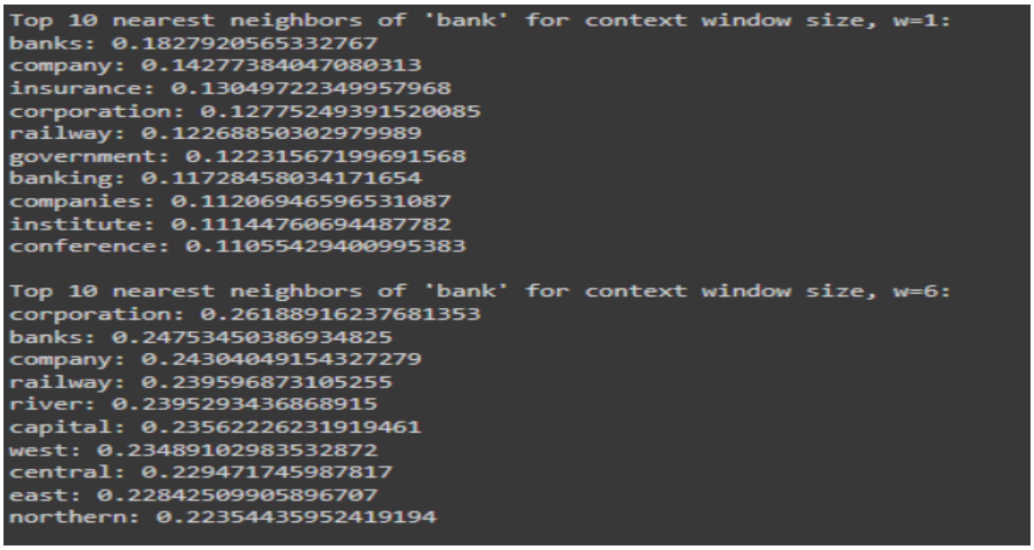
Overall:

The nearest neighbors of a word tend to match its part of speech tags, especially with smaller window sizes. Larger windows introduce more context-driven relationships, particularly for adjectives and prepositions. Nouns and verbs exhibit more consistent nearest neighbors across different window sizes.

5.3) Now try choosing words with multiple senses (e.g., bank, cell, apple, apples, axes, frame, light, well, etc.) as query words. What appears to be happening with multisense words based on the nearest neighbors that you observe? What happens when you compare the neighbors with different window 6 sizes (w = 1 vs. w = 6)? Discuss your findings, showing examples of nearest neighbors for particular words to support your claims.

For different multisense words:

1)bank

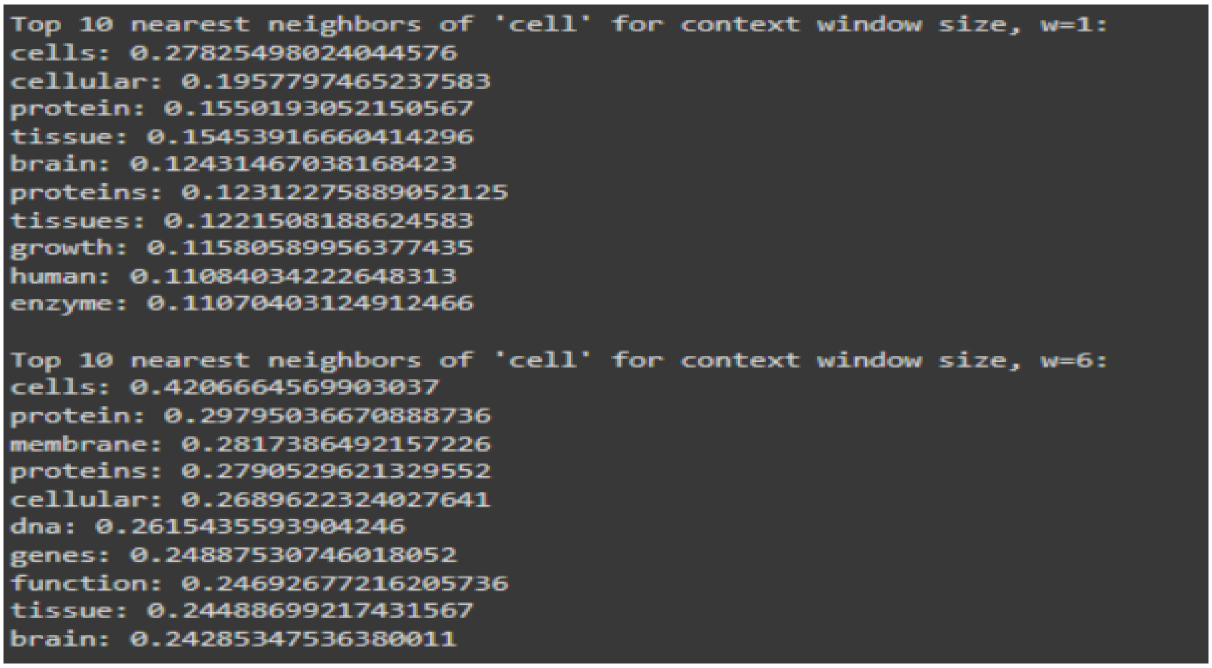


```
Top 10 nearest neighbors of 'bank' for context window size, w=1:
banks: 0.1827920565332767
company: 0.14277384047080313
insurance: 0.13049722349957968
corporation: 0.12775249391520085
railway: 0.1226885030297989
government: 0.12231567199691568
banking: 0.11728458034171654
companies: 0.11206946596531087
institute: 0.11144760694487782
conference: 0.11055429400995383

Top 10 nearest neighbors of 'bank' for context window size, w=6:
corporation: 0.26188916237681353
banks: 0.24753450386934825
company: 0.24304049154327279
railway: 0.239596873105255
river: 0.2395293436868915
capital: 0.23562226231919461
west: 0.23489102983532872
central: 0.229471745987817
east: 0.22842509905896707
northern: 0.22354435952419194
```

Window size w = 1, neighbors like "banks", "company", and "insurance", relate mainly to the financial sense of the word. The small window highlights the local context, emphasizing the narrower sense of "bank" as a financial institution.

Window size w = 6, neighbors here like "river" and "capital" appear, indicating a broader geographical sense of "bank" like riverbank. The larger window captures multiple meanings, with "river" suggesting the geographical interpretation, and "capital" connecting to both finance and geography.

2)cell

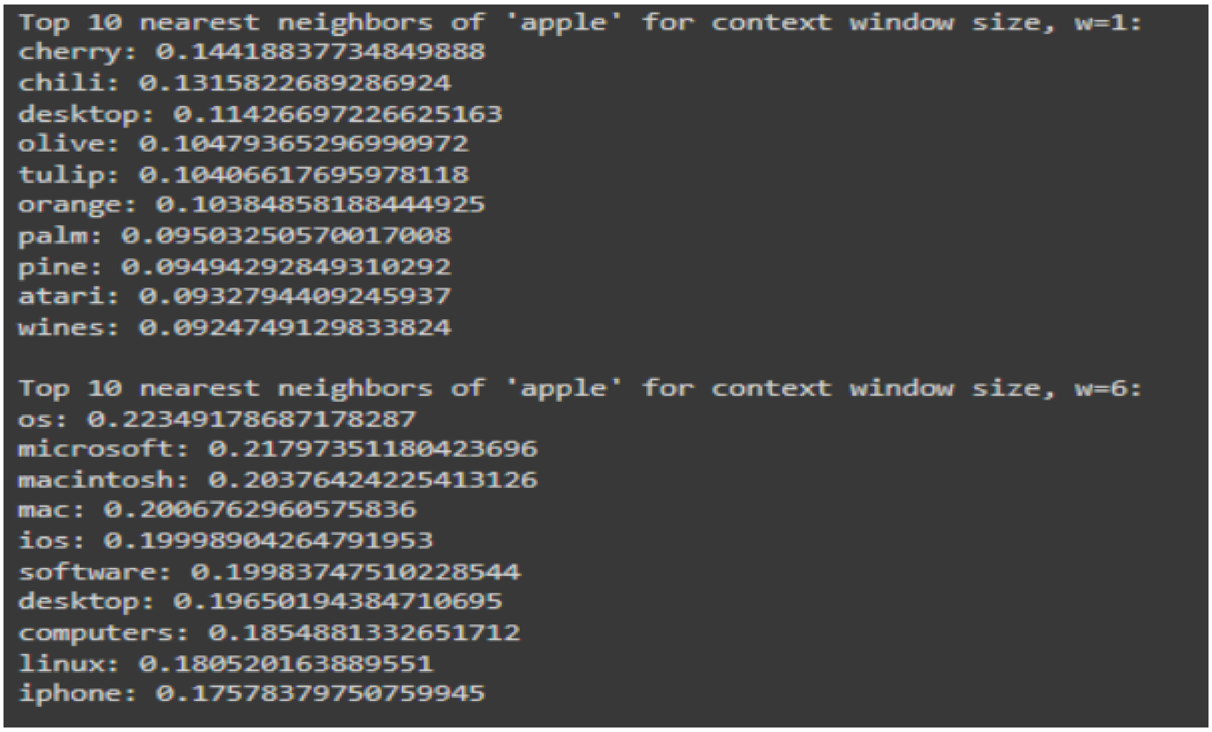


```
Top 10 nearest neighbors of 'cell' for context window size, w=1:
cells: 0.2782549802404576
cellular: 0.1957797465237583
protein: 0.1550193052150567
tissue: 0.15453916660414296
brain: 0.12431467038168423
proteins: 0.12312275889052125
tissues: 0.1221508188624583
growth: 0.11580589956377435
human: 0.11084034222648313
enzyme: 0.11070403124912466

Top 10 nearest neighbors of 'cell' for context window size, w=6:
cells: 0.4206664569903037
protein: 0.29795036670888736
membrane: 0.2817386492157226
proteins: 0.2790529621329552
cellular: 0.2689622324027641
dna: 0.2615435593904246
genes: 0.24887530746018052
function: 0.24692677216205736
tissue: 0.24488699217431567
brain: 0.24285347536380011
```

Window size w = 1, neighbors like "cells", "cellular", "protein" etc. suggest a strong biological sense.

Window size w = 6, neighbors like "cells", "membrane", "proteins", "dna", "genes" etc. maintains the biological theme but introduces broader related concepts like "dna" and "genes," reflecting a more abstract understanding of cellular biology.

3)apple

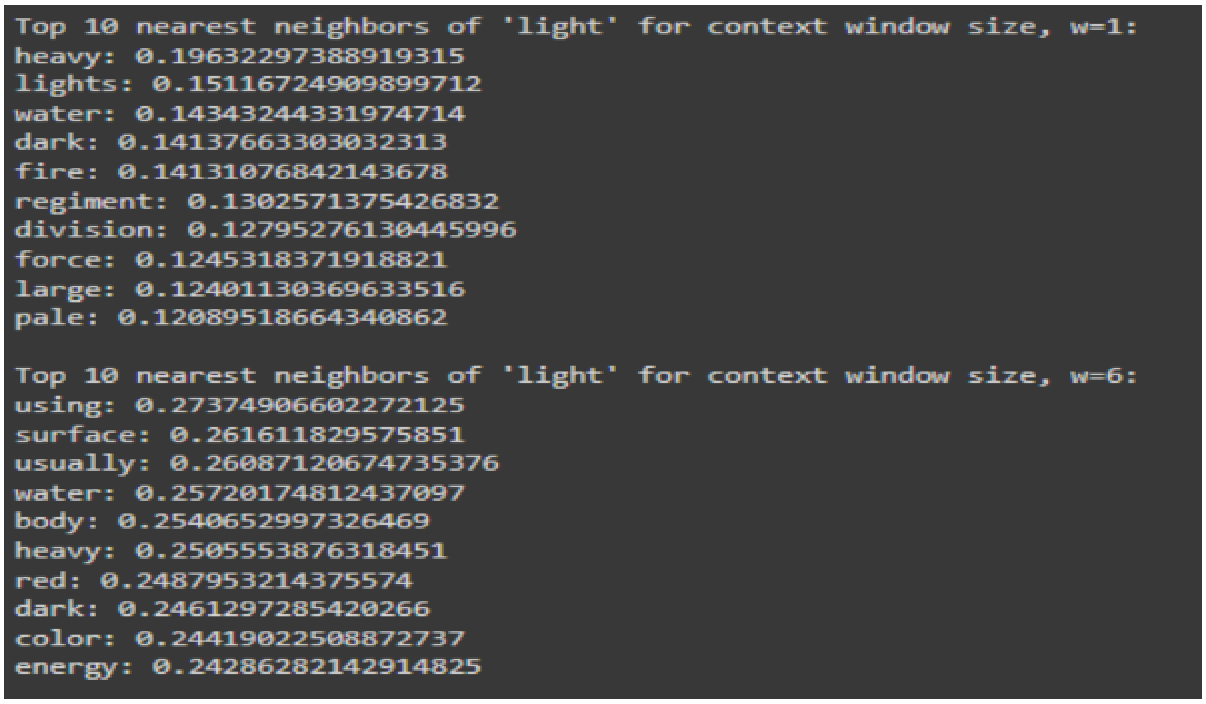


```
Top 10 nearest neighbors of 'apple' for context window size, w=1:
cherry: 0.14418837734849888
chili: 0.1315822689286924
desktop: 0.11426697226625163
olive: 0.10479365296990972
tulip: 0.10406617695978118
orange: 0.10384858188444925
palm: 0.09503250570017008
pine: 0.09494292849310292
atari: 0.0932794409245937
wines: 0.0924749129833824

Top 10 nearest neighbors of 'apple' for context window size, w=6:
os: 0.22349178687178287
microsoft: 0.21797351180423696
macintosh: 0.20376424225413126
mac: 0.2006762960575836
ios: 0.19998904264791953
software: 0.19983747510228544
desktop: 0.19650194384710695
computers: 0.1854881332651712
linux: 0.180520163889551
iphone: 0.17578379750759945
```

Window size w = 1, neighbors like "cherry", "chili", "olive", “orange” etc. have food-related senses of "apple", with neighbors like "cherry," "olive," and "orange." It also includes "desktop" which highlights at a technical meaning.

Window size w = 6, neighbors like "os", "microsoft", "macintosh", "mac" etc. clearly shifts to the technology-related sense of "apple" focusing on the technology domain.

4)light



```
Top 10 nearest neighbors of 'light' for context window size, w=1:
heavy: 0.19632297388919315
lights: 0.15116724909899712
water: 0.14343244331974714
dark: 0.14137663303032313
fire: 0.14131076842143678
regiment: 0.1302571375426832
division: 0.12795276130445996
force: 0.1245318371918821
large: 0.12401130369633516
pale: 0.12089518664340862

Top 10 nearest neighbors of 'light' for context window size, w=6:
using: 0.27374906602272125
surface: 0.261611829575851
usually: 0.26087120674735376
water: 0.25720174812437097
body: 0.2540652997326469
heavy: 0.2505553876318451
red: 0.2487953214375574
dark: 0.2461297285420266
color: 0.24419022508872737
energy: 0.24286282142914825
```

Window size w = 1, neighbors like "heavy", "lights", "water", "dark" etc. focus on physical characteristics associated with light.

Window size w = 6, neighbors like "using", "surface", "water”, "body", "energy" etc. focus toward a broader scientific sense, highlighting a physics-related sense of "light".

Overall:

1) I noticed that window size impacts the nearest neighbors for words with multiple senses based on PMI vectors as shown in the examples above. Words with multiple senses have varied meanings, which makes them perfect candidates for studying the effect of different window sizes.
2) Smaller window sizes like w = 1 tend to emphasize literal meanings of words by focusing on immediate neighboring words. This is useful for capturing concrete senses, such as "bank" in a financial context or "apple" as a fruit.
3) Larger window sizes like w = 6 expand the scope to include words that co-occur in a broader context, which introduces general senses of words. For example, "apple" in the tech sense becomes dominant, and for "light" scientific meanings are captured.
4) For multisense words, smaller window sizes stick onto one meaning, while larger windows unveil different senses. This shift is visible across words like "bank," "apple," and "light," where the nearest neighbors differ significantly based on window size.