1. Word Embeddings:
    a. Can you find the top-10 similar words to wonderful and awful? Do your results make sense?

        Top-10 similar to 'wonderful':
        [('wonderful', 0.9999998807907104), ('marvelous', 0.8188857436180115), ('fantastic', 0.8047919869422913), ('great', 0.7647868394851685), ('fabulous', 0.7614760994911194), ('terrific', 0.7420832514762878), ('lovely', 0.7320097088813782), ('amazing', 0.7263179421424866), ('beautiful', 0.6854087114334106), ('magnificent', 0.6633867025375366)]

        Top-10 similar to 'awful':
        [('awful', 1.0), ('horrible', 0.7597668170928955), ('terrible', 0.7478912472724915), ('dreadful', 0.7218177914619446), ('horrid', 0.6720177531242371), ('atrocious', 0.6626646518707275), ('ugly', 0.6236302256584167), ('lousy', 0.6135217547416687), ('unbelievable', 0.6068726181983948), ('appalling', 0.6061565279960632)]

        i. These words make sense to me. They seem similar

    b. Also come up with 3 other queries and show your results. Do they make sense?

        Top-10 similar to 'far':
        [('far', 1.0000001192092896), ('much', 0.5893325805664062), ('farther', 0.4790627956390381), ('so', 0.45866602659225464), ('well', 0.45197173953056335), ('yet', 0.4362397789955139), ('just', 0.4346715211868286), ('none', 0.422846257686615), ('many', 0.4186747074127197), ('than', 0.41073712706565857)]

        Top-10 similar to 'pale':
        [('pale', 1.0), ('dim', 0.45742517709732056), ('grayish', 0.4542086124420166), ('sickly', 0.45420700311660767), ('wispy', 0.4423399567604065), ('plump', 0.44080761075019836), ('gaunt', 0.43972039222717285), ('radiant', 0.43631094694137573), ('pasty', 0.43369346857070923), ('gray', 0.42931845784187317)]

        Top-10 similar to 'young':
        [('young', 1.0), ('teenage', 0.6431925296783447), ('younger', 0.635094165802002), ('teenaged', 0.6070023775100708), ('impressionable', 0.5947335362434387), ('youth', 0.5715782642364502), ('youngsters', 0.5662700533866882), ('teenagers', 0.5633072853088379), ('talented', 0.5397676229476929), ('youthful', 0.5372710824012756)]

        i. The “far”, I would say, is a bit subjective and does not make sense.

    c. Find top 10 words closest to the following two queries. Do your results make sense? (sister - woman + man and harder - hard + fast)

Top-10 similar to 'sister - women + man':
[('sister', 0.6695989370346069), ('brother', 0.664355993270874), ('nephew', 0.6285309195518494), ('son', 0.6248186230659485), ('uncle', 0.6227600574493408), ('daughter', 0.586700439453125), ('mother', 0.5849319696426392), ('grandmother', 0.5821883678436279), ('father', 0.5821642279624939), ('boy', 0.5701210498809814)]

Top-10 similar to 'harder - hard + fast':
[('faster', 0.7290612459182739), ('fast', 0.7240180373191833), ('harder', 0.5723087191581726), ('easier', 0.5631422400474548), ('rapidly', 0.4939379096031189), ('slow', 0.47127774357795715), ('tougher', 0.4532335102558136), ('bigger', 0.4532044231891632), ('smarter', 0.4506851136684418), ('cheaper', 0.44532695412635803)]

i. They mostly make sense except they do add some that can be considered negative such as female based nouns in the 'sister-women+man'

d. Also come up with 3 other queries and show your results. Do they make sense?

Top-10 similar to 'king - man + women':
[('king', 0.6478992700576782), ('queen', 0.535493791103363), ('women', 0.5233659148216248), ('kings', 0.5162314176559448), ('queens', 0.4995364248752594), ('monarch', 0.45280295610427856), ('royals', 0.4143899381160736), ('princesses', 0.39867541193962097), ('kingdom', 0.38632214069366455), ('princess', 0.382215678691864)]

Top-10 similar to 'walking - walk + run':
[('run', 0.6202627420425415), ('running', 0.6052120923995972), ('walking', 0.5376755595207214), ('ran', 0.4823693633079529), ('runs', 0.43421316146850586), ('chasing', 0.36862990260124207), ('coasting', 0.3641245663166046), ('doing', 0.360645592212677), ('riding', 0.3600478768348694), ('pulling', 0.3587511479854584)]

Top-10 similar to 'doctor - hospital + school':
[('school', 0.594773530960083), ('teacher', 0.5409976840019226), ('doctor', 0.5294280052185059), ('schoolers', 0.48368263244628906), ('elementary', 0.44890716671943665), ('kid', 0.4261430501937866), ('parents', 0.4260135889053345), ('dentist', 0.4205280840396881), ('kids', 0.4190119802951813), ('students', 0.4188379645347595)]

i. For the first one, it is mostly good though there are still a few male type words that should be more negative. The last one seems to stray a little with "kids" but is generally good with the academic setting.

2. Better Perceptron using Embeddings

a. For the first sentence in the training set (+), find a different sentence in the training set that is closest to it in terms of sentence embedding. Does it make sense in terms of meaning and label?

   Original (+) sentence: {1.0000}: 'it 's a tour de force , written and directed so quietly that it 's implosion rather than explosion you fear'

   Closest sentence: {0.7771}: 'a semi autobiographical film that 's so sloppily written and cast that you can not believe anyone more central to the creation of bugsy than the caterer had anything to do with it'

   i. I suppose there are similarities with them being written films though their contents and purposes are different.

b. For the second sentence in the training set (-), find a different sentence in the training set that is closest to it in terms of sentence embedding. Does it make sense in terms of meaning and label?

   Original: {1.0000}: 'exactly what you 'd expect from a guy named kaos'

   Closest: {0.9121}: 'it 's exactly what you 'd expect'

   i. Very similar, almost the same sentence.

c. Report the error rates of k-NN classifier on dev for k = 1, 3, ...99 using sentence embedding. You can reuse your code from HW1/HW2 and/or use sklearn. (should be ~28%).

- Loading data...
- Loading embeddings...
- Training KNN model...
  - K = 1, Dev Err = 0.393
  - K = 3, Dev Err = 0.366
  - K = 5, Dev Err = 0.339
  - K = 7, Dev Err = 0.321
  - K = 9, Dev Err = 0.325
  - K = 11, Dev Err = 0.334
  - K = 13, Dev Err = 0.338
  - K = 15, Dev Err = 0.339
  - K = 17, Dev Err = 0.331
  - K = 19, Dev Err = 0.314
  - K = 21, Dev Err = 0.319
  - K = 23, Dev Err = 0.309
  - K = 25, Dev Err = 0.310
  - K = 27, Dev Err = 0.316
  - K = 29, Dev Err = 0.312
  - K = 31, Dev Err = 0.310
  - K = 33, Dev Err = 0.312
  - K = 35, Dev Err = 0.305

- K = 37, Dev Err = 0.303
- K = 39, Dev Err = 0.289
- K = 41, Dev Err = 0.293
- K = 43, Dev Err = 0.301
- K = 45, Dev Err = 0.304
- K = 47, Dev Err = 0.304
- K = 49, Dev Err = 0.301
- K = 51, Dev Err = 0.294
- K = 53, Dev Err = 0.290
- K = 55, Dev Err = 0.298
- K = 57, Dev Err = 0.295
- K = 59, Dev Err = 0.297
- K = 61, Dev Err = 0.295
- K = 63, Dev Err = 0.295
- K = 65, Dev Err = 0.287
- K = 67, Dev Err = 0.291
- K = 69, Dev Err = 0.286
- K = 71, Dev Err = 0.284
- K = 73, Dev Err = 0.288
- K = 75, Dev Err = 0.292
- K = 77, Dev Err = 0.288
- K = 79, Dev Err = 0.291
- K = 81, Dev Err = 0.286
- K = 83, Dev Err = 0.288
- K = 85, Dev Err = 0.292
- K = 87, Dev Err = 0.285
- K = 89, Dev Err = 0.284
- K = 91, Dev Err = 0.289
- K = 93, Dev Err = 0.289
- K = 95, Dev Err = 0.293
- K = 97, Dev Err = 0.294
- K = 99, Dev Err = 0.291

- Predicting with KNN...
- Writing output...
  - Err Rate 0.284

d. Report the error rates of k-NN classifier on dev for k = 1, 3, ...99 using one-hot vectors from HW2. You can reuse your code from HWs 1-2 and/or use sklearn. (should be ~40%).
   - Loading data...
   - Loading embeddings...
   - Training one-hot KNN (pipeline)...
     - K = 1, Dev Err = 0.402
     - K = 3, Dev Err = 0.406

- K = 5, Dev Err = 0.414
- K = 7, Dev Err = 0.418
- K = 9, Dev Err = 0.407
- K = 11, Dev Err = 0.391
- K = 13, Dev Err = 0.391
- K = 15, Dev Err = 0.383
- K = 17, Dev Err = 0.402
- K = 19, Dev Err = 0.401
- K = 21, Dev Err = 0.392
- K = 23, Dev Err = 0.399
- K = 25, Dev Err = 0.396
- K = 27, Dev Err = 0.400
- K = 29, Dev Err = 0.393
- K = 31, Dev Err = 0.399
- K = 33, Dev Err = 0.398
- K = 35, Dev Err = 0.413
- K = 37, Dev Err = 0.415
- K = 39, Dev Err = 0.404
- K = 41, Dev Err = 0.401
- K = 43, Dev Err = 0.400
- K = 45, Dev Err = 0.405
- K = 47, Dev Err = 0.411
- K = 49, Dev Err = 0.412
- K = 51, Dev Err = 0.406
- K = 53, Dev Err = 0.400
- K = 55, Dev Err = 0.410
- K = 57, Dev Err = 0.426
- K = 59, Dev Err = 0.410
- K = 61, Dev Err = 0.409
- K = 63, Dev Err = 0.403
- K = 65, Dev Err = 0.410
- K = 67, Dev Err = 0.416
- K = 69, Dev Err = 0.413
- K = 71, Dev Err = 0.409
- K = 73, Dev Err = 0.406
- K = 75, Dev Err = 0.400
- K = 77, Dev Err = 0.397
- K = 79, Dev Err = 0.407
- K = 81, Dev Err = 0.406
- K = 83, Dev Err = 0.394
- K = 85, Dev Err = 0.399
- K = 87, Dev Err = 0.410
- K = 89, Dev Err = 0.418
- K = 91, Dev Err = 0.412

- K = 93, Dev Err = 0.398
- K = 95, Dev Err = 0.395
- K = 97, Dev Err = 0.403
- K = 99, Dev Err = 0.400

- Best K = 15, Dev Err = 0.383
- Predicting with one-hot KNN...
- Writing output...
  - Err Rate 0.383

e. Submit your best k-NN classifier to Kaggle and report the public error rate and ranking (take a screenshot). (should be ~26%).

89 43865  0.270

f. For basic perceptron, show the training logs for 10 epochs. Compare your best dev error rate (should be ~31%) with the one from HW2.
   i.
   - Training percept model…
     - Epoch = 1, Dev Err = 0.374
     - Epoch = 2, Dev Err = 0.255
     - Epoch = 3, Dev Err = 0.273
     - Epoch = 4, Dev Err = 0.274
     - Epoch = 5, Dev Err = 0.300
     - Epoch = 6, Dev Err = 0.340
     - Epoch = 7, Dev Err = 0.268
     - Epoch = 8, Dev Err = 0.266
     - Epoch = 9, Dev Err = 0.241
     - Epoch = 10, Dev Err = 0.431
     - Err 0.241 vs 37.7%
   - Predicting with percept...

   ii.

g. For averaged perceptron, show the training logs for 10 epochs. Compare your best dev error rate (should be ~23–24%) with the one from HW2.
   i.
   - Training avg percept model…
     - Epoch = 1, Dev Err = 0.260
     - Epoch = 2, Dev Err = 0.246
     - Epoch = 3, Dev Err = 0.251
     - Epoch = 4, Dev Err = 0.241
     - Epoch = 5, Dev Err = 0.234
     - Epoch = 6, Dev Err = 0.247
     - Epoch = 7, Dev Err = 0.246
     - Epoch = 8, Dev Err = 0.246
     - Epoch = 9, Dev Err = 0.239
     - Epoch = 10, Dev Err = 0.239
     - Err 0.234 vs 26.3%

- Predicting with avg percept...

h. Do you need to use smart averaging here?
   i. Smart averaging seems to yield better results though only slightly

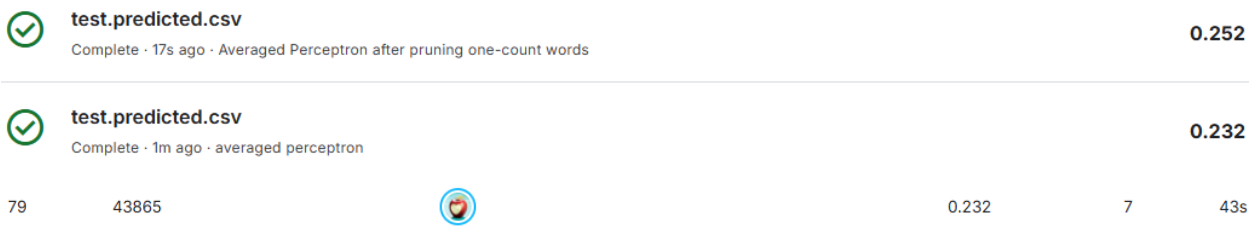
i. For averaged perceptron after pruning one-count words, show the training logs for 10 epochs. Compare your dev error rate (should be ~23.5%) with the one from HW2.
   i.
   - Training Averaged One-Count Perceptron...
     - One-count pruning: pruned=7370, kept=8411
     - Epoch 1, Dev Err = 0.234
     - Epoch 2, Dev Err = 0.241
     - Epoch 3, Dev Err = 0.234
     - Epoch 4, Dev Err = 0.229
     - Epoch 5, Dev Err = 0.231
     - Epoch 6, Dev Err = 0.225
     - Epoch 7, Dev Err = 0.229
     - Epoch 8, Dev Err = 0.229
     - Epoch 9, Dev Err = 0.234
     - Epoch 10, Dev Err = 0.237
     - Best Dev Err: 0.225
     - Predicting...

j. For the above setting, give at least two examples on dev where using features of word2vec is correct but using one-hot representation is wrong, and explain why.
   i. Words: 'provoking' vs 'escapism'
      Word2Vec similarity: 0.1273
      One-hot similarity: 0.0000

   ii. Words: 'thriller' vs 'drama'
      Word2Vec similarity: 0.6441
      One-hot similarity: 0.0000

   iii. The words have similarities in word2vec but are not similar enough for one-hot representation. This is because for word2vec, it uses dimensions for word context similarity while one-hot only checks if they are the same position coding.

k. Submit your averaged perceptron models (both with and without pruning) to Kaggle, and record best your public error rate and ranking (take a screenshot). (should be ~22.4%)



3. Try some other learning algorithms with sklearn
    a. Which algorithm did you try? What adaptations did you make to your code to make it work with that algorithm?
        i. SVM I already had a variety of data forms for the sentences so it was fairly easy to “plug and play” to call the LinearSVC and using the 300 dim vectors.

    b. What’s the dev error rate(s) and running time?
        - Training Perceptron...
            - Best Dev Err: 0.293
            - Training time: 0.02s
        - Training Averaged Perceptron...
            - Best Dev Err: 0.238
            - Training time: 0.03s
        - Training Averaged One-Count Perceptron...
            - Dropped 7456, Kept 8352
                - Best Dev Err: 0.247
            - Training time: 0.34s
        - Training KNN...
            - Best K = 7
                - Dev Err = 0.321
            - Training time: 2.61s
        - Training One-Hot KNN...
            - Best K = 1
                - Dev Err = 0.402
            - Training time: 1.56s
        - Training SVM...
            - Best Dev Err: 0.231
            - Training time: 0.34s

    c. What did you learn in terms of the comparison between averaged perceptron and these other (presumably more popular and well-known) learning algorithms?

        i. Averaged percepteron is pretty good as a fast and dirty algorithm. Though the SVM handles more complex data and takes considerably longer. If I were to test with a larger set or with less clean data then SVM would have a clearer distinction in its effectiveness.

4. Deployment
    a. What's your best error rate on dev, and which algorithm and setting achieved it?
        i. My best was SVM using the standard embedding settings. Dev_err = 0.231

    b. What's your best public error rate on Kaggle, and which algorithm and setting achieved it? Part of your grade depends on your public and private error rates. For example, our TA got 21%.
        i. My best was an averaged perceptron of 23.6% which was the same as a general SVM later.

5. Debriefing
    a. Approximately how many hours did you spend on this assignment?
        i. 12 hours
    b. Would you rate it as easy, moderate, or difficult?
        i. Moderate: it was able to reuse a lot of old code but I tried to improve or go simpler and make my overall program more modular which took time while i was battling holidays, sickness and work.
    c. Did you work on it mostly alone, or mostly with other people?
        i. Alone.
    d. How deeply do you feel you understand the material it covers (0%–100%)?
        i. 40% I read the embedding which I believe were the main point of the assignment, and implemented it. Though most of my time working was resetting the previous knn and perceptron models themselves and plugging in embedded values.
    e. Any other comments?
        i. Im tired captain