Chimdi Chikezie
AI534/Fall2023
Homework 4

Part 1: Word Embeddings

Q: Can you find the top 10 similar words to wonderful and awful?

wv.most_similar('awful', topn=10)	**wv.most_similar('student', topn=10)**
[('horrible', 0.7597668170928955), ('terrible', 0.7478912472724915), ('dreadful', 0.7218177914619446), ('horrid', 0.6720177531242371), ('atrocious', 0.6626645922660828), ('ugly', 0.6236302256584167), ('lousy', 0.6135217547416687), ('unbelievable', 0.6068726181983948), ('appalling', 0.6061565279960632), ('hideous', 0.5811460614204407)]	[('students', 0.7294867038726807), ('teacher', 0.6301366090774536), ('school', 0.6055627465248108), ('undergraduate', 0.6020305752754211), ('university', 0.600540041923523), ('campus', 0.5629045367240906), ('academic', 0.5484224557876587), ('professors', 0.530981183052063), ('college', 0.525564968585968), ('grad', 0.5203014016151428)]

Do your results make sense?

The words make sense because, according to the English dictionary, they are close in meaning to awful and student.

Q: Also come up with 3 other queries and show your results.

wv.most_similar('family', topn=10)
[('students', 0.7294867038726807), ('teacher', 0.6301366090774536), ('school', 0.6055627465248108), ('undergraduate', 0.6020305752754211), ('university', 0.600540041923523), ('campus', 0.5629045367240906), ('academic', 0.5484224557876587), ('professors', 0.530981183052063), ('college', 0.525564968585968), ('grad', 0.5203014016151428)]
wv.most_similar('success', topn=10)
[('successes', 0.724018394947052), ('successful', 0.6167578101158142), ('accomplishment', 0.49159350991249084),

('achievements', 0.4895521104335785), ('achievement', 0.4850277304649353), ('triumphs', 0.4617437720298767), ('greatness', 0.45542895793914795), ('progress', 0.44958776235580444), ('popularity', 0.4400866627693176), ('triumph', 0.43484923243522644)]
wv.most_similar('enhance', topn=10)
[('enhancing', 0.7954033613204956), ('improve', 0.7549501657485962), ('enhances', 0.7072708010673523), ('enhanced', 0.6955755352973938), ('bolster', 0.6009522080421448), ('elevate', 0.5607604384422302), ('enable', 0.5488600134849548), ('boost', 0.5353639721870422), ('reduce', 0.5311101675033569), ('fortify', 0.5228351354598999)]

Do they make sense?

The words in the output make sense because, according to the English dictionary, they are close in meaning to the words in the code.

Q: Find top 10 words closest to the following two queries.

sister - woman + man	harder - hard + fast
[('brother', 0.7966989874839783), ('uncle', 0.6753759980201721), ('nephew', 0.6596081852912903), ('son', 0.6472460031509399), ('father', 0.6398823261260986)]	[('faster', 0.7064898610115051), ('rapidly', 0.5021133422851562), ('easier', 0.48843103647232056), ('slow', 0.4575234949588756), ('quickly', 0.4370785653591156)]

Do your results make sense?

The output shows that the model has effectively learned certain aspects of words.

Example: It correctly identifies brother as the most closely related term to sister when the gender is shifted towards man. The other terms, while not as directly equivalent, are still contextually relevant and show a good understanding of the semantics involved in royalty and gender.

Q: Also come up with 3 other queries and show your results.

wv.most_similar(positive=['king', 'woman'], negative=['man'], topn=5)
[('queen', 0.7118193507194519), ('monarch', 0.6189674139022827), ('princess', 0.5902431011199951), ('kings', 0.5236844420433044), ('queens', 0.5181134343147278)]
wv.most_similar(positive=['winter', 'hot'], negative=['cold'], topn=5)
[('summer', 0.5669187903404236), ('spring', 0.5186282396316528), ('hottest', 0.5085405707359314), ('summertime', 0.4723301827907562), ('season', 0.4058019518852234)]
wv.most_similar(positive=['doctor', 'law'], negative=['medicine'], topn=5)
[('laws', 0.4727955460548401), ('lawyer', 0.4094833433628082), ('judge', 0.38079601526260376), ('legally', 0.3534584939479828), ('dentist', 0.34965550899505615)]

Do they make sense?

The output shows that the model has effectively learned certain aspects of words.

Example: It correctly identifies 'queen' as the most closely related term to 'king' when the gender is shifted towards 'woman'. The other terms, while not as directly equivalent, are still contextually relevant and show a good understanding of the semantics involved in royalty and gender.

Part 2: Better k-NN and Perceptron using Embeddings

2.1 Reimplement k-NN with Sentence Embedding

1. 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.

First senetence: it 's a tour de force , written and directed so quietly that it 's implosion rather than explosion you fear.

Closest sentence label: -
Closest sentence: 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.

Does it make sense in terms of meaning and label?
Yes, it makes sense. The sentence has some negative words and the label was predicted to be negative which is not same for the main sentence.

2. 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.
Main sentence: places a slightly believable love triangle in a difficult to swallow setting , and then disappointingly moves the story into the realm of an improbable thriller.

Closest sentence label: -
Closest sentence: the plan to make enough into an inspiring tale of survival wrapped in the heart pounding suspense of a stylish psychological thriller ' has flopped as surely as a souffl gone wrong.

Does it make sense in terms of meaning and label?
Yes, it makes sense. The sentence has some negative words and the label was also predicted to be negative which is the same as the main sentence..

3. Report the error rate of k-NN classifier on dev for k = 1, 3, ...99 using sentence embedding. You can reuse your code from HW1 or use sklearn.

Best k: 73
smallest dev error: 0.278

4. Report the error rate 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 around 40%).

Best k: 15
smallest dev error: 0.383

2.2 Reimplement Perceptron with Sentence Embedding

1. For basic perceptron, show the training logs for 10 epochs (should be around 26 33%). Compare your error rate with the one from HW2.

Hw 4	*Hw2*
epoch 1, update 31.1%, dev 37.4% epoch 2, update 29.5%, dev 35.4% epoch 3, update 29.8%, dev 33.2% epoch 4, update 29.1%, dev 40.0% epoch 5, update 29.7%, dev 35.2% epoch 6, update 29.4%, dev 40.4% epoch 7, update 29.4%, dev 38.4% epoch 8, update 29.4%, dev 42.5% epoch 9, update 29.1%, dev 39.0% epoch 10, update 29.1%, dev 39.2% best dev err 33.2%, |w|=300, time: 5.8 secs	epoch 1, update 39.0%, dev 39.6% epoch 2, update 25.5%, dev 34.1% epoch 3, update 20.8%, dev 35.3% epoch 4, update 17.2%, dev 35.5% epoch 5, update 14.1%, dev 28.9% epoch 6, update 12.2%, dev 32.0% epoch 7, update 10.5%, dev 32.0% epoch 8, update 9.7%, dev 31.5% epoch 9, update 7.8%, dev 30.2% epoch 10, update 6.9%, dev 29.8% best dev err 28.9%, |w|=16744, time: 1.2 secs

2. For averaged perceptron, show the training logs for 10 epochs (should be around 23%). Compare your error rate with the one from HW2.

Hw 4	*Hw2*
epoch 1, update 31.1%, dev 24.9% epoch 2, update 29.5%, dev 23.9% epoch 3, update 29.8%, dev 24.3% epoch 4, update 29.1%, dev 24.1% epoch 5, update 29.7%, dev 24.2% epoch 6, update 29.4%, dev 23.9% epoch 7, update 29.4%, dev 23.6% epoch 8, update 29.4%, dev 23.8% epoch 9, update 29.1%, dev 24.1% epoch 10, update 29.1%, dev 24.4% best dev err 24.4%, |w|=300, time: 7.5 secs	epoch 1, update 39.0%, dev 31.4% epoch 2, update 25.5%, dev 27.7% epoch 3, update 20.8%, dev 27.2% epoch 4, update 17.2%, dev 27.6% epoch 5, update 14.1%, dev 27.2% epoch 6, update 12.2%, dev 26.7% epoch 7, update 10.5%, dev 26.3% epoch 8, update 9.7%, dev 26.4% epoch 9, update 7.8%, dev 26.3% epoch 10, update 6.9%, dev 26.3% best dev err 26.3%, |w|=15806, time: 1.5 secs

3. Do you need to use smart averaging here?

Yes, it gave us a better dev error.

4. For averaged perceptron after pruning one-count words, show the training logs for 10 epochs. Compare your error rate with the one from HW2.

5. 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. (1 pt)

Part 3: Try some other learning algorithms with sklearn

1. Approximately how many hours did you spend on this assignment?
72

1. Which algorithm did you try?
I tried logistic regression and decision trees.

What adaptations did you make to your code to make it work with that algorithm?
I Converted sentences to numerical data using TfidfVectorizer from sklearn.feature_extraction.text

2. What's the dev error rate(s) and running time?
Logistic Regression dev error: 0.274
Logistic Regression runtime: 0.1319589614868164 seconds
Decision Tree dev error: 0.42500000000000004
Decision Tree runtime: 2.638833999633789 seconds

3. What did you learn in terms of the comparison between averaged perceptron and these other (presumably more popular and well-known) learning algorithms?
Averaged Perceptron is simple and efficient in linear classification tasks, but may underperform on complex, non-linear datasets compared to more flexible algorithms like Logistic Regression and Decision Trees, which handle non-linear relationships better. The choice of algorithm depends on the balance between interpretability, complexity, and the specific nature of the data.

Part 4: Deployment

Q: what's your best error rate on dev, and
24.4%

which algorithm and setting achieved it?
averaged perceptron using sentence embeddings

Part 5: Debriefing

1. Approximately how many hours did you spend on this assignment?
72

2. Would you rate it as easy, moderate, or difficult?
Moderate

3. Did you work on it mostly alone, or mostly with other people?
Alone

4. How deeply do you feel you understand the material it covers (0%–100%)?
90

5. Any other comments?
I love that I gained real-world knowledge