Latent Semantic Analysis

Yuhuan Huang, Aug 2025

1.SVD

Console Output:

```
>>> np.isclose(reconstructed[1], [0.05308057,-0.2712466,0.76690292,1.
array([ True,  True,  True,  True,  True])
>>> np.isclose(error, 0.7801, atol=1e-03)
np.True_
"""

octest.testmod()

[10]  ✓  0.0s

...  U:
 [[-0.6611152   0.49727948 -0.56181831]
 [-0.2260912  -0.84604119 -0.48280128]
 [-0.7154086  -0.19216509  0.6717612 ]]
Sigma:
 [2.27249817 1.49235868 0.78013949]
Vt:
 [[-0.60573153 -0.29092001 -0.09949016 -0.41430167 -0.60573153]
 [ 0.20445111  0.33321713 -0.56691545 -0.69568147  0.20445111]
 [ 0.14092723 -0.7201511  -0.61886533  0.24221299  0.14092723]]
Reconstructed (k=2):
 [[ 1.06176793  0.68436019 -0.2712466   0.10616114  1.06176793]
 [ 0.05308057 -0.2712466   0.76690292  1.09123009  0.05308057]
 [ 0.92614463  0.37740774  0.32432717  0.87306406  0.92614463]]

...  TestResults(failed=0, attempted=6)
```

Reflection:

1. Describe in your own words what the 'Original Document-Term Matrix' represents in your code output. Be specific about what each row, column, and individual value represents in the context of movie reviews.

My Ans: In the document-term matrix, a row represents a review, and a column represents a word. The element represents the existence of the word in the review.

2. Analyze the U matrix output. How do the values in this matrix relate to the sentiment of each review? Provide specific examples of how different reviews might be grouped together based on these patterns.

   My Ans: The matrix U describes how strongly each review exhibits the pattern. From the U matrix, we can see that the first columns are all negative, this means that all of the three reviews are negative on the first pattern(component); And for the second column, we know that the second and the third review are negative, while the first review is positive on the second pattern; As for the third column, we can know that the first and the second review are negative on the third pattern, while the third review is positive on the third pattern.

3. Examine the singular values output. What do these values tell us about the importance of different patterns in our movie reviews? Why might some patterns be more significant than others?

   My Ans: The singular value represents the strength of each pattern. Since the eigen values are: 2.2725, 1.4926, 0.8701, we can see that the first pattern is the strongest, whereas the third pattern is the weakest. Some possible reason that some patterns are more significant than others: 1. Some words are just be more frequently used in the review. 2. Similar types of words tend to appear together in a review.

4. Study the V^T matrix output. How do the values show relationships between words in our vocabulary? Identify specific word groupings or patterns that might represent concepts like "positive" or "negative" sentiment.

   My Ans: Each element $V_{i,j}$ of the V transpose matrix shows how strongly word j loads on component i. Within the same row, words with the same sign and larger absolute values tend to co-occur. Therefore, from the V matrix, we can see that according to the second row, words "action", "exciting", and "amazing" should be grouped together.

5. When we reduce dimensions to k=2, what information is preserved and what is lost? Support your answer by comparing specific values between the original and reconstructed matrices.

My Ans: Since we have reduced the dimension to k=2, from the reconstruction matrix we can see that the information of the third pattern is not well-preserved.

6. Examine the reconstruction error. How does changing the value of k affect this error, and what does this tell us about the trade-off between dimensionality reduction and information preservation? Provide specific examples from your experimentation.

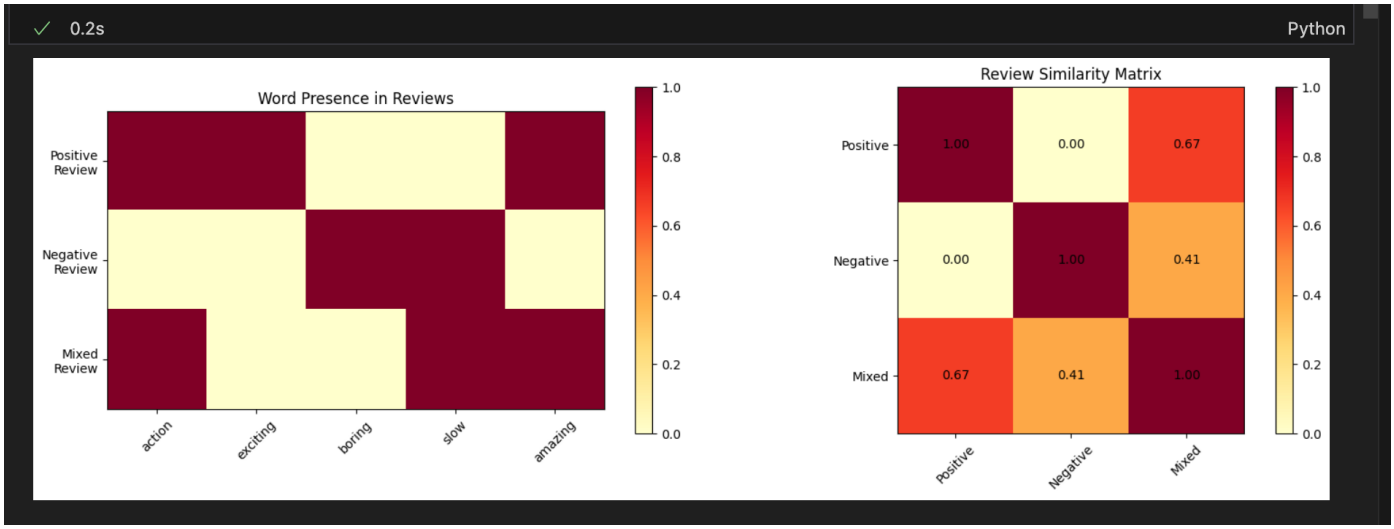
My Ans: When k=2, the reconstruction error is 0.7801, and if we try k=1, that is reduce even more dimension, the reconstruction error becomes 1.6840, then the reconstruction error increases.

```
Reconstruction error:
 1.6839691391292742
```

So we can see the tradeoff: with the increase of dimensional reduction, the error also increases.

2.Cosine Similarity

Console output:



```
...
    Cosine Similarity Analysis:

    Review Comparisons:

    Positive Review vs Negative Review:
    Cosine Similarity: 0.000
    Shared words: None

    Positive Review vs Mixed Review:
    Cosine Similarity: 0.667
    Shared words: action, amazing

    Negative Review vs Mixed Review:
    Cosine Similarity: 0.408
    Shared words: slow

...   TestResults(failed=0, attempted=3)
```

Reflection:

1. Consider the following scenario:
   review_A = [1, 1, 1, 0, 0] # "action exciting boring"
   review_B = [0, 0, 0, 1, 1] # "slow amazing"
   Despite having no words in common, these reviews could theoretically have a non-zero cosine similarity. Explain why this is false and use the cosine similarity formula to prove your reasoning. What does this tell us about the relationship between shared terms and similarity scores?

   My Ans: Using the Cosine Similarity eqn:

$$\cos\theta = \frac{A \cdot B}{\|A\| \, \|B\|}$$

   It is false because, since $A \cdot B$ is 0, the Cosine similarity should also be 0. So if there's no shared term, the cosine similarity should be 0.

2. You are analyzing a large dataset of movie reviews and find two reviews with the following vectors:
   review_1 = [2, 0, 3, 1, 0] # Word counts instead of binary presence
   review_2 = [4, 0, 6, 2, 0] # Doubled values of review_1
   Without calculating, what would their cosine similarity be? Explain why this value makes sense in terms of what cosine similarity measures, and how this differs from

Euclidean distance. How might this property be useful when comparing documents of different lengths?

My Ans: The Cosine similarity should be 1, since the two vectors are exactly in the same direction. It is different from the Euclidean distance since the Euclidean distance takes the length into account. It is useful since we can directly measure the similarity without having to consider the impact of the length of vector.

3. Given three movie reviews:

   review_X = [1, 1, 0, 0, 0] # "action exciting"
   review_Y = [0, 1, 0, 0, 1] # "exciting amazing"
   review_Z = [1, 0, 0, 0, 1] # "action amazing"

   If the cosine similarity between X and Y is equal to the cosine similarity between X and Z, what does this tell us about the relative importance of the words "exciting" and "amazing" in this vector space? What are the limitations of using binary vectors (0s and 1s) instead of weighted term frequencies in this case?

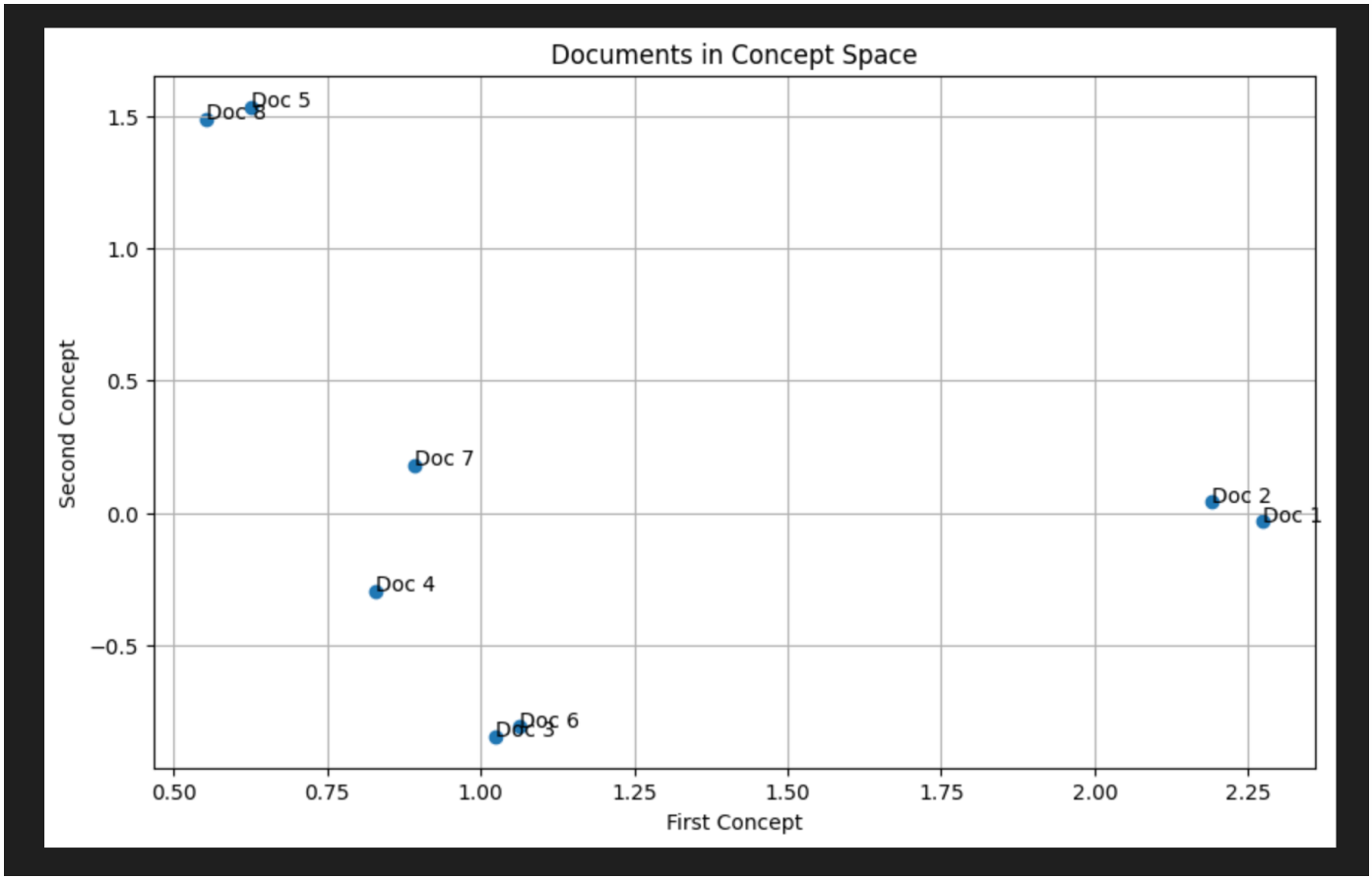
My Ans: It means that the two words have equal importance influence on the words. If we only use 0/1, but not the frequency, we would lose the frequency similarity, but only preserve the existence.

3. LSA Implementation

Console Output:

```
...   Document-Term Matrix (Word Counts):
          brown  bush  cat  climbs  dog  forest  fox  garden  hops  hunts  jumps  \
    Doc 1      1     0    0       0    1       0    1       0     0      0      1
    Doc 2      1     0    1       0    1       0    0       0     0      0      1
    Doc 3      0     0    0       0    0       0    1       0     0      0      0
    Doc 4      1     0    1       1    0       0    0       0     0      0      0
    Doc 5      0     0    0       0    0       1    0       0     0      0      0
    Doc 6      0     1    0       0    1       0    0       0     0      0      0
    Doc 7      1     0    0       0    0       1    1       0     0      1      0
    Doc 8      0     0    0       0    0       0    0       1     1      0      0

          lazy  quick  rabbit  runs  sleeps  tree
    Doc 1     1      1       0     0       0     0
    Doc 2     1      1       0     0       0     0
    Doc 3     1      0       0     0       1     1
    Doc 4     0      0       0     0       0     1
    Doc 5     0      1       1     1       0     0
    Doc 6     1      0       0     0       1     0
    Doc 7     0      0       0     0       0     0
    Doc 8     0      1       1     0       0     0


    Concept 1 important words:
      lazy: 0.457
      brown: 0.432
    ...
    Doc 7   0.892432   0.181261
    Doc 8   0.553377   1.490581
```

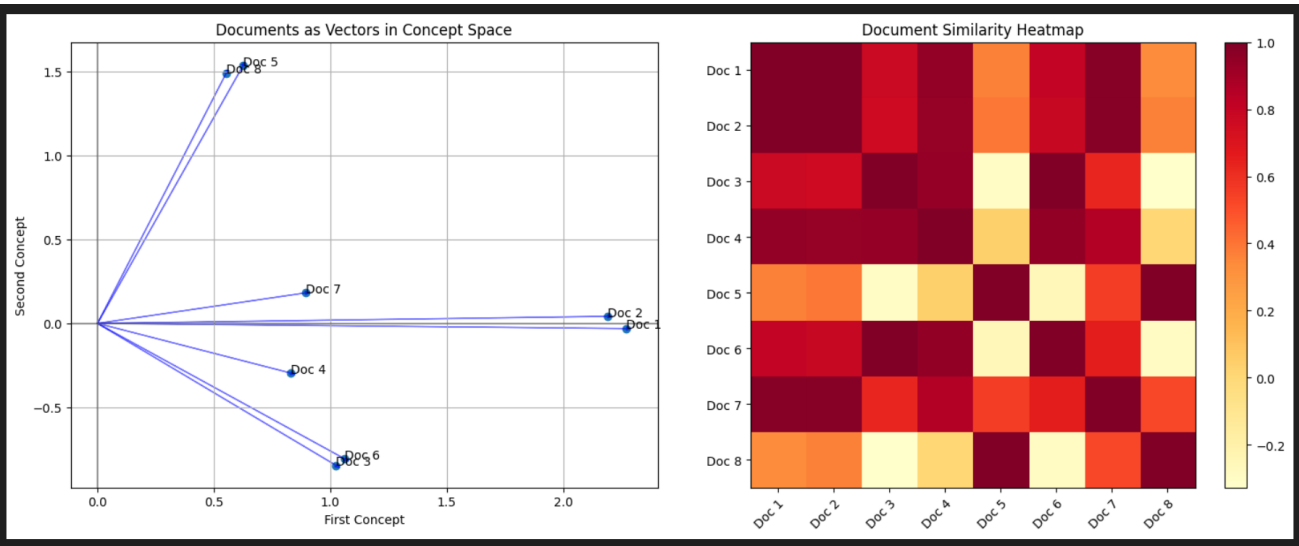
```
Concept 1 important words:
  lazy: 0.457
  brown: 0.432
  quick: 0.394
  dog: 0.386
  jumps: 0.312

Concept 2 important words:
  quick: 0.501
  rabbit: 0.499
  forest: 0.283
  sleeps: -0.272
  lazy: -0.270

Document-Concept Matrix:
       Concept 1  Concept 2
Doc 1   2.272492  -0.030753
Doc 2   2.190909   0.041995
Doc 3   1.024848  -0.845405
Doc 4   0.829986  -0.295504
Doc 5   0.625771   1.535146
Doc 6   1.062852  -0.806992
Doc 7   0.892432   0.181261
Doc 8   0.553377   1.490581
```



```python
#For test validation - do not change any code below this line
import doctest
"""
    >>> doc_term_df.shape[0] == len(documents)
    True
    >>> doc_concept_df.shape == (len(documents), 2)
    True
    >>> len(concept_terms) == 2
    True
    >>> len(similar_docs) == len(documents) - 1
    True
    >>> print(concept_terms[0][0][0])
    lazy
    >>> np.isclose(concept_terms[0][0][1], 0.4572429831217709, atol=1e-03)
    np.True_
    >>> print(concept_terms[1][0][0])
    quick
    >>> np.isclose(concept_terms[1][0][1], 0.500544036009478, atol=1e-03)
    np.True_
    >>> print(concept_terms[0][3][0])
    dog
    >>> np.isclose(concept_terms[0][3][1], 0.3857123074245487, atol=1e-03)
    np.True_
    >>> print(concept_terms[1][4][0])
    lazy
    >>> np.isclose(concept_terms[1][4][1], -0.2704899548034848, atol=1e-03)
    np.True_


"""
doctest.testmod()
```

✓ 0.0s

```
...
    Documents similar to document 1:
    Query document: 'The quick brown fox jumps over the lazy dog'
    Document 2: 0.999 similarity
    Text: 'A quick brown dog jumps over the lazy cat'
    Document 7: 0.977 similarity
    Text: 'The brown fox hunts in the forest'
    Document 4: 0.947 similarity
    Text: 'A brown cat climbs up the tree'
    Document 6: 0.805 similarity
    Text: 'A lazy dog sleeps under the bush'
    Document 3: 0.780 similarity
    Text: 'The lazy fox sleeps under the tree'
    Document 5: 0.365 similarity
    Text: 'The quick rabbit runs through the forest'
    Document 8: 0.335 similarity
    Text: 'A quick rabbit hops through the garden'
    For doctest - Concept 1 terms: [('lazy', np.float64(0.45724298312177103)), ('brown', np.float64(0.431747538975484)), ('quick',
    For doctest - Concept 2 terms: [('quick', np.float64(0.5005440360094779)), ('rabbit', np.float64(0.4986910936040569)), ('forest

...   TestResults(failed=0, attempted=12)
```




```
Angle Analysis for Document 1:
Angle between Doc 1 and Doc 2: 1.9° (similarity: 0.999)
Angle between Doc 1 and Doc 3: 38.7° (similarity: 0.780)
Angle between Doc 1 and Doc 4: 18.8° (similarity: 0.947)
Angle between Doc 1 and Doc 5: 68.6° (similarity: 0.365)
Angle between Doc 1 and Doc 6: 36.4° (similarity: 0.805)
Angle between Doc 1 and Doc 7: 12.3° (similarity: 0.977)
Angle between Doc 1 and Doc 8: 70.4° (similarity: 0.335)
```

Reflection:

1. In the visualization, we observed that documents can appear far apart in the concept space plot but still have high cosine similarity. Explain this phenomenon and discuss why cosine similarity was chosen as the similarity measure for LSA rather than Euclidean distance. Support your answer with specific examples from the results.

   My Ans: The length matters a lot in the scatter points, even when the angle is small, a long distance can still result in far-apart points. But consine similarity only cares about the angle. An example is doc 1 and doc 4, the angle is only 18.8 degrees, with a similarity of 0.977, but they seem to be far from each other.

2. Compare the importance of words in Concept 1 versus Concept 2. What semantic patterns do you observe? For example, do certain types of words (e.g., actions, subjects, locations) tend to dominate different concepts? What does this tell us about how LSA captures meaning?

My Ans: In concept 1, the important words are "lazy", "brown", "quick", "dog", and the semantic pattern is like: subject, desciptors, action, modifier. In concept 2, the important words are "quick", "rabbit", "forest", "sleepy", "lazy", and the semantic pattern is also like: subject, descriptors, action, modifier. I think the desciptors dominate both concepts. We can know that LSA captures meaning by co-occurrence of words.

3. Several documents containing the word "lazy" appear in the collection. However, LSA allows us to find semantic relationships beyond simple word matching. Analyze how LSA groups these documents compared to traditional keyword matching. Does the presence of "lazy" guarantee high similarity between documents? Why or why not?

My Ans: No, it is possible thjat the same word "lazy" has different positive or negative in the context. For example, we can say that the "dog" is "lazy", then it is possibly positive; however, in another context when we say the "worker" is lazy, it is possibly negative.

4. Changing the number of concepts (n_components) in LSA can significantly impact results. If we increased n_components from 2 to 3, what trade-offs would we face? Consider both computational aspects and the interpretability of results. How would you determine the optimal number of concepts for a given document collection?

My Ans: If we increase our n_components, we would have better (higher accuracy) interpretation of the result. However, the computational load would be heavier. To select the optimal number of n_components, we can compare the increase in accuracy and loss in efficiency. We can also use some criterias such as the increase in information entropy or HIC, etc. to find the "best" n_components.

5. The implementation removes common stop words before performing LSA. Analyze how the choice of stop words affects the final results. Find a specific example in the results where including or excluding a particular stop word might change the semantic relationships captured by LSA. Would keeping all stop words necessarily make the results worse? Explain your reasoning.

My Ans: In this example, some prepositions are chosen as the stop words. However, sometimes prepositions matter. For example, if we are introducing our house to others, like "there is a storage box on the table", "there is a storage box under the table", "there is a storage box next to the table". In this scenario, the prepositions matter, and it may matter if we keep using the prepositions as the stop words.

```
stop_words = ['the', 'a', 'and', 'over', 'under', 'through', 'in', 'up']
```