Worksheet 11

Name: Rishab Sudhir

UID: U64819615

Topics

- Latent Semantic Analysis

Latent Semantic Analysis

In this section we will fetch news articles from 3 different categories. We will perform Tfidf vectorization on the corpus of documents and use SVD to represent our corpus in the feature space of topics that we've uncovered from SVD. We will attempt to cluster the documents into 3 clusters as we vary the number of singular vectors we use to represent the corpus, and compare the output to the clustering created by the news article categories. Do we end up with a better clustering the more singular vectors we use?

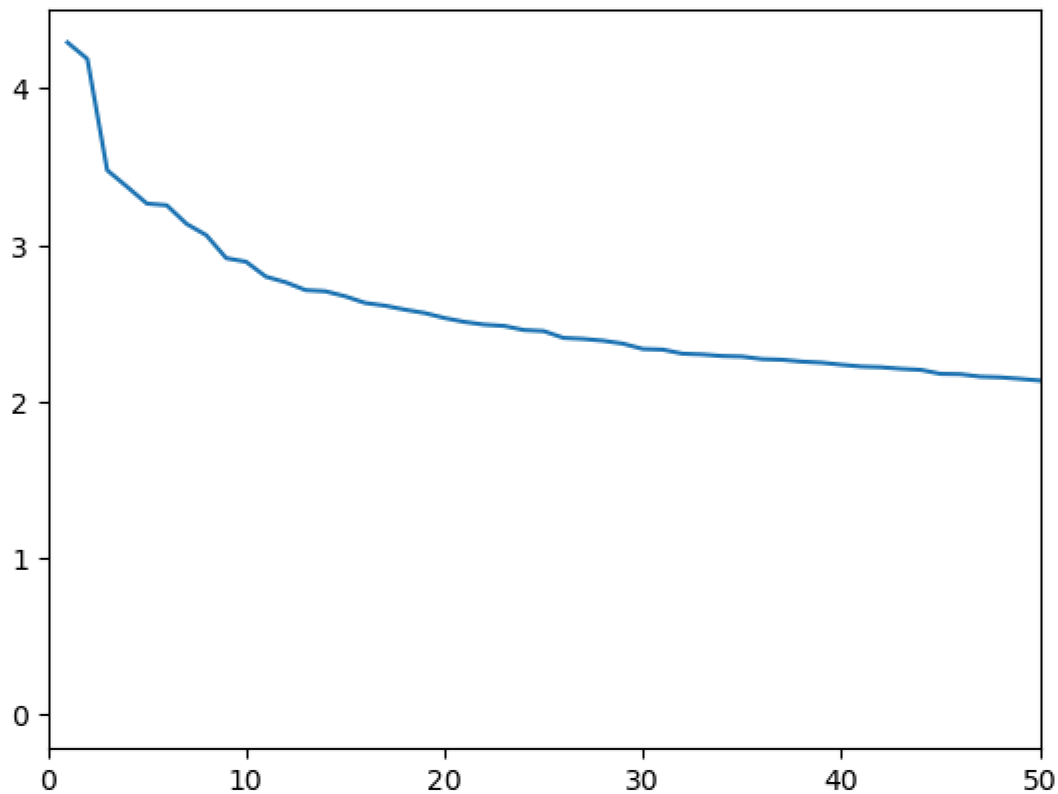
In [8]:
```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
```

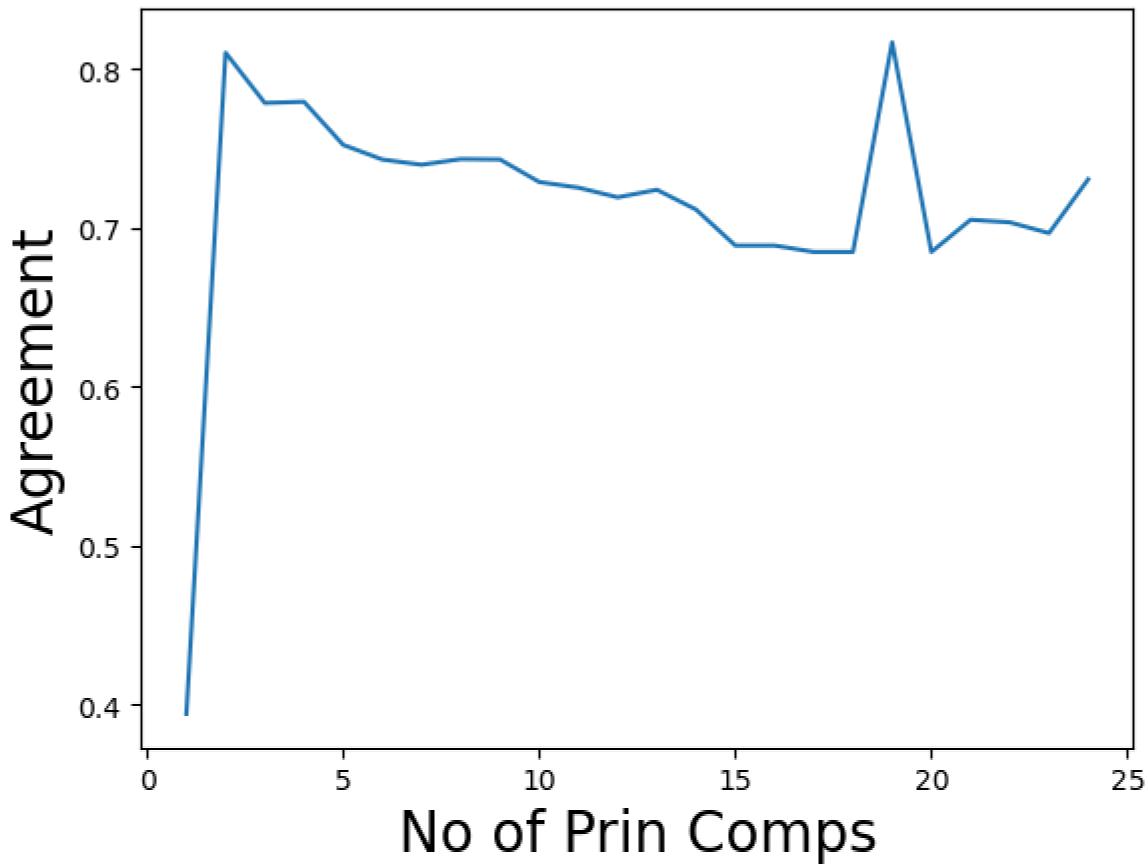
```
[nltk_data] Downloading package punkt to /Users/rsudhir/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/rsudhir/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[8]:
```
True
```

In [9]:
```
import numpy as np
from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.stem.snowball import SnowballStemmer
from nltk.tokenize import word_tokenize, sent_tokenize

# Categories of news
categories = ['comp.os.ms-windows.misc', 'sci.space','rec.sport.basebal

# Fetching those cartegories as training subsets
news_data = fetch_20newsgroups(subset='train', categories=categories)

# Reduces words to their root, eg skiing, skied, would become ski
# Also removes stopwords such as the, a, I etc
stemmed_data = [" ".join(SnowballStemmer("english", ignore_stopwords=Tr
         for sent in sent_tokenize(message)
        for word in word_tokenize(sent))
        for message in news_data.data]

# Used to transform preprocessed text int a TF-IDF matrix
# considering words that appear in at least 4 documents (min_df=4)
# excluding words that appear in more than 80% of the documents (max_di
# Stop words in English are also removed to focus on meaningful content
vectorizer = TfidfVectorizer(stop_words='english', min_df=4,max_df=0.8)

# Document-Term Matrix(DTM) The DTM is centered by subtracting the mear
# of each term's frequency across all documents. This step is
# crucial for SVD, as it focuses on the variance around the mean.
dtm = vectorizer.fit_transform(stemmed_data)
terms = vectorizer.get_feature_names_out()
centered_dtm = dtm - np.mean(dtm, axis=0)

# SVD
u, s, vt = np.linalg.svd(centered_dtm)

# Plotting the Singular Values
plt.xlim([0,50])
plt.plot(range(1,len(s)+1),s)
plt.show()

# The code iterates through a range of singular vectors (1 to 24) to cr
# clusters using K-means. For each number of singular vectors, it trans
# the documents into the reduced space defined by those vectors and app
# K-means clustering with 3 clusters, aiming to mirror the original cat

ag = []
max = len(u)
for singular_vectors in range(1,25):
    vectorsk = u.dot(np.diag(s))[:,:singular_vectors]
    kmeans = KMeans(n_clusters=3, init='k-means++', max_iter=100, n_ini
    kmeans.fit_predict(np.asarray(vectorsk))
    labelsk = kmeans.labels_
    ag.append(metrics.v_measure_score(labelsk, news_data.target))
```

```
# The disagreement distance measure or v measure score
# The quality of clustering is evaluated using the V-measure,
# a harmonic mean of completeness and homogeneity scores, which compare
# the clustering labels with the actual categories of the documents.
# The results are plotted to show how the agreement (similarity to actu
# categories) changes as the number of principal components (singular v
# used varies.

# closer to 1 means closer to news categories

# (if points are in the same cluster they are similar)
plt.plot(range(1,25),ag)
plt.ylabel('Agreement',size=20)
plt.xlabel('No of Prin Comps',size=20)
plt.show()
```





Does the best at 3 components for the 3 documents. Which is ideal as there are three categories

Explaining the steps of the code above:

The code explores the effectiveness of Latent Semantic Analysis (LSA) through its components —TF-IDF vectorization and Singular Value Decomposition (SVD)— for document clustering, and specifically, how the number of singular vectors (dimensions) influences the quality of clustering in comparison to actual news article categories.

1. **Fetch News Articles**: You'll need a dataset of news articles categorized into at least three different categories. This dataset will serve as the basis for your analysis.
2. **Preprocess the Text**: Before applying TF-IDF, preprocess the text to clean and normalize it. This includes removing stop words, stemming or lemmatization, and possibly filtering out punctuation and numbers, depending on your specific goals.
3. **TF-IDF Vectorization**: Transform the cleaned text documents into a TF-IDF matrix. TF-IDF stands for Term Frequency-Inverse Document Frequency, a numerical statistic that reflects how important a word is to a document in a collection or corpus. This step converts the text data into a format suitable for mathematical analysis.
4. **Apply SVD**: Perform Singular Value Decomposition on the TF-IDF matrix to reduce its dimensionality while preserving its most significant information. This process uncovers the latent topics (features) within the documents. The number of singular vectors (dimensions) you choose to keep will influence the results of your clustering.

5. **Document Clustering**: With the documents now represented in the reduced feature space, you can apply clustering algorithms (such as K-means) to group them into clusters. Since you know the actual categories, you aim to see if the clustering algorithm can uncover similar groupings based on the document contents rather than predefined labels.
6. **Varying Number of Singular Vectors**: Experiment with different numbers of singular vectors to represent your corpus and perform clustering for each case. This step is crucial to understanding how the choice of dimensionality affects the quality of the clusters.
7. **Evaluation and Comparison**: Finally, compare the clusters obtained from LSA + clustering with the actual categories of the news articles. You can use metrics such as purity, NMI (Normalized Mutual Information), or Rand index to quantify the similarity between the clusters and the actual categories.

Key Questions:

- **Do we end up with better clustering the more singular vectors we use?** This depends. Initially, adding more singular vectors (increasing dimensionality) might improve clustering by capturing more nuanced relationships between documents. However, beyond a certain point, adding more dimensions could introduce noise or overfitting, potentially leading to worse clustering performance.
- **What's the optimal number of singular vectors?** This is highly dataset-specific. You'll

Embeddings

The data comes from the [Yelp Dataset (https://www.yelp.com/dataset)](https://www.yelp.com/dataset). Each line is a review that consists of a label (0 for negative reviews and 1 for positive reviews) and a set of words.

```
1 i will never forget this single breakfast experience in ma
d...
0 the search for decent chinese takeout in madison continues
...
0 sorry but me julio fell way below the standard even for me
d...
1 so this is the kind of food that will kill you so there s
t...
```

In order to transform the set of words into vectors, we will rely on a method of feature engineering called word embeddings (Tfidf is one way to get these embeddings). Rather than simply indicating which words are present, word embeddings represent each word by "embedding" it in a low-dimensional vector space which may carry more information about the semantic meaning of the word. (for example in this space, the words "King" and "Queen" would be close).

`word2vec.txt` contains the `word2vec` embeddings for about 15 thousand words. Not every word in each review is present in the provided `word2vec.txt` file. We can treat these words as being "out of vocabulary" and ignore them.

Example

Let x_i denote the sentence `“a hot dog is not a sandwich because it is not square”` and let a toy word2vec dictionary be as follows:

```
hot      0.1     0.2     0.3
not      -0.1    0.2     -0.3
sandwich 0.0     -0.2    0.4
square   0.2     -0.1    0.5
```

we would first `trim` the sentence to only contain words in our vocabulary: `"hot not sandwich not square”` then embed x_i into the feature space:

$$\varphi 2(x_i)) = \frac{1}{5}(word2vec(\text{hot}) + 2 \cdot word2vec(\text{not}) + word2vec(\text{sandwich}) \\ + word2vec(\text{square})) = [0.02 \ \ 0.06 \ \ 0.12 \ \ ]^T$$

a) Implement a function to trim out-of-vocabulary words from the reviews. Your function

In [13]:
```
import csv
import numpy as np

VECTOR_LEN = 300   # Length of word2vec vector
MAX_WORD_LEN = 64  # Max word length in dict.txt and word2vec.txt

def load_tsv_dataset(file):
    """
    Loads raw data and returns a tuple containing the reviews and their

    Parameters:
        file (str): File path to the dataset tsv file.

    Returns:
        An np.ndarray of shape N. N is the number of data points in the
        Each element dataset[i] is a tuple (label, review), where the l
        an integer (0 or 1) and the review is a string.
    """
    dataset = np.loadtxt(file, delimiter='\t', comments=None, encoding=
                         dtype='l,O')
    return dataset


def load_feature_dictionary(file):
    """
    Creates a map of words to vectors using the file that has the word2
    embeddings.

    Parameters:
        file (str): File path to the word2vec embedding file.

    Returns:
        A dictionary indexed by words, returning the corresponding word
        embedding np.ndarray.
    """
    word2vec_map = dict()
    with open(file) as f:
        read_file = csv.reader(f, delimiter='\t')
        for row in read_file:
            word, embedding = row[0], row[1:]
            word2vec_map[word] = np.array(embedding, dtype=float)
    return word2vec_map


def trim_reviews(path_to_dataset):

    # Load the dataset and word2vec dictionary
    dataset = load_tsv_dataset(path_to_dataset)
    feature_dict = load_feature_dictionary("./data/word2vec.txt")

    # Initialize an empty list to store the trimmed reviews
    trimmed_reviews = []

    for label, review in dataset:
        # Split the review into words
        words = review.split(" ")
        # Keep only the words that are in the feature dictionary (word2
```

```
        trimmed_words = [word for word in words if word in feature_dict

        trimmed_review = " ".join(trimmed_words)

        # Append the trimmed review and its label as a tuple to the tri
        trimmed_reviews.append((label, trimmed_review))

    # Convert the list of trimmed reviews back into a NumPy array with
    return np.array(trimmed_reviews, dtype=dataset.dtype)

print("Un-Trimmed version - Train")
print(load_tsv_dataset("./data/train_small.tsv")[1])
trim_train = trim_reviews("./data/train_small.tsv")
print("Trimmed Version")
print(trim_train[1])


print("Un-Trimmed version - Test")
print(load_tsv_dataset("./data/test_small.tsv")[1])
trim_test = trim_reviews("./data/test_small.tsv")
print("Trimmed Version")
print(trim_test[1])
```

```
Un-Trimmed version - Train
(0, 'the search for decent chinese takeout in madison continues the f
ood at this place we ordered delivery is nothing worth waiting 1 5 ho
urs for greasy vegetable egg rolls which contain nothing but cabbage 
greasy lo mein they clearly do not understand how to work with tofu n
or do they get how to deliver quality food in a timely fashion the on
e plus was the vegetables were still crunchy just save your time and 
money and don t bother not even a good dousing of sriracha can make 
their ho hum food sparkle the reviews i ve seen from in house diners 
do not provide any contrast to our delivery experience')
Trimmed Version
(0, 'the search for decent chinese in madison continues the food at t
his place we ordered delivery is nothing worth waiting hours for egg 
rolls which contain nothing but lo they clearly do not understand how
to work with nor do they get how to deliver quality food in a timely 
fashion the one plus was the were still just save your time and money
and don t bother not even a good of can make their ho food the review
s i ve seen from in house do not provide any contrast to our delivery
experience')
Un-Trimmed version - Test
(0, 'we came for the live music which was great i love that they have
bands there on the weekends but the food and drinks weren t great i h
ad a margarita which was way too sweet i sent it back and they remade
it the same they are pre mixed so there s no room for change the thin
g is in madison there s an incredible craft cocktail scene so the com
petition is stiff and many of us don t want an overly sweet pre mixed
drink but a legit tangy margarita the chips and salsa were also medio
cre compared to places like eldorado grill tex tubbs laredo s i d try
it one more time for dinner in a pinch perhaps after they ve had a li
ttle more time to grow')
Trimmed Version
(0, 'we came for the live music which was great i love that they have
bands there on the but the food and drinks t great i had a which was 
way too sweet i sent it back and they it the same they are mixed so t
here s no room for change the thing is in madison there s an incredib
le craft cocktail scene so the competition is stiff and many of us do
n t want an overly sweet mixed drink but a the chips and were also me
diocre compared to places like s i d try it one more time for dinner 
in a perhaps after they ve had a little more time to grow')
```