Word Vectors using Truncated SVD on Amazon Fine Food Review Dataset

Exercise :

1. Download Amazon Fine Food Reviews dataset from Kaggle. You may have to create a Kaggle account to download data. (https://www.kaggle.com/snap/amazon-fine-food-reviews)
2. Perform featurization - tf-idf.
3. Separate top 2000, 5000 and 10000 features on the basis of their TFIDF score, respectively.
4. Also calculate co-occurrence matrix of separated dataset.
5. Apply TruncatedSVD on the co-occurrence matrix with optimal n_component value.(calculate n*component(k) using explained* variance_ratio)
6. Perform k-means to cluster the dataset, and evaluate whether words are semantically similar or not.
7. Write your observations in English as crisply and unambiguously as possible. Always quantify your results.

Information regarding data set :

1. **Title**: Amazon Fine Food Reviews Data
2. **Sources**: Stanford Network Analysis Project(SNAP)
3. **Relevant Information**: This dataset consists of reviews of fine foods from amazon. The data span a period of more than 10 years, including all ~568,454 reviews up to October 2012(Oct 1999 - Oct 2012). Reviews include product and user information, ratings, and a plain text review.
4. **Attribute Information**:
   ProductId - unique identifier for the product
   UserId - unqiue identifier for the user
   ProfileName - name of the user
   HelpfulnessNumerator - number of users who found the review helpful
   HelpfulnessDenominator - number of users who indicated whether they found the review helpful or not
   Score - rating between 1 and 5.(rating of 4 or 5 could be cosnidered a positive review. A review of 1 or 2 could be considered negative. A review of 3 is nuetral and ignored)
   Time - timestamp for the review
   Summary - brief summary of the review
   Text - text of the review

Objective :

> Analyze, transform data into TF-IDF vectorizers and apply TruncatedSVD to relate the Matrix Factorization technique to find top-n word vectors in the dataset.Also perform k-means++ algorithm to cluster them.

In [1]:
```
import warnings
from sklearn.exceptions import DataConversionWarning
warnings.filterwarnings(action='ignore', category=DataConversionWarning)
warnings.filterwarnings(action='ignore', category=UserWarning)
warnings.filterwarnings(action='ignore', category=FutureWarning)

import traceback
import sqlite3
import itertools
import pandas as pd
import numpy as np
import datetime as dt
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud, STOPWORDS
from collections import OrderedDict
from tqdm import tqdm
from prettytable import PrettyTable
from sklearn import preprocessing
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.cluster import KMeans
from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import normalize
```

Load dataset :

In [2]:
```
# This dataset is already gone through data deduplication and text preprocessing, so it is approx ~364
K

# For Data Cleaning Steps follow this link -
# ipython notebook - https://drive.google.com/open?id=1JXCva5vXdIPgHbfNdD9sgnySqELoVtpy
# dataset - https://drive.google.com/open?id=1IoDoTT8TfDu53N6cyKg6xVCU-FDPHyIF

# For Text Preporcessing Steps follow this link -
# ipython notebook - https://drive.google.com/open?id=18-AkTzzEhCwM_hflIbDNBMAP-imX4k4i
# dataset - https://drive.google.com/open?id=1SfDwwXFhDpjgtfIE5O_E80SO89xRc8Sa

# Load dataset
def load_review_dataset(do_not_sample=True, sample_count=1):
    # Create connection object to load sqlite dataset
    connection = sqlite3.connect('finalDataSet.sqlite')

    # Load data into pandas dataframe.
    reviews_df = pd.read_sql_query(""" SELECT * FROM Reviews """,connection)

    # Drop index column
    reviews_df = reviews_df.drop(columns=['index'])

    # Sample dataset
    if do_not_sample == False:
        reviews_df = reviews_df.sample(sample_count)

    # Convert timestamp to datetime.
    reviews_df['Time'] = reviews_df[['Time']].applymap(lambda x: dt.datetime.fromtimestamp(x))

    # Sort the data on the basis of time.
    reviews_df = reviews_df.sort_values(by=['Time'])

    return reviews_df


# Load 'finalDataSet.sqlite' in panda's daraframe.
reviews_df = load_review_dataset(do_not_sample = True,sample_count = 1)


# Make CleanedText as a dataset for clustering
CLEANED_TEXT = reviews_df['CleanedText'].values

print("Dataset Shape : \n",CLEANED_TEXT.shape)
```

```
Dataset Shape :
 (351237,)
```

Perform Featurization on Text Data :

Term Frequency - Inverse Document Frequency (TF-IDF) :

In [3]:
```
def perform_tfidf(reviews,max_features):
    # Instantiate TfidfVectorizer
    tfidf_vectorizer = TfidfVectorizer(min_df = 3,ngram_range=(1,1),max_features=max_features)

    # Tokenize and build vocab
    tfidf_vectorizer.fit(reviews)

    # Encode document
    tfidf_dtm = tfidf_vectorizer.transform(reviews)

    print("\nThe type of tfidf vectorizer ",type(tfidf_dtm))
    print("The shape of data matrix ",tfidf_dtm.get_shape())
    print("The number of unique words in data matrix ", tfidf_dtm.get_shape()[1])

    # Data Normalization
    # tfidf_dtm = preprocessing.normalize(tfidf_dtm)

    return tfidf_dtm,tfidf_vectorizer
```

Let's Separate top 2000, 5000 and 10000 features from TfidfVectorizer :

In [6]:
```
# Perform TFIDF Featurization on raw data
# Max 2000 features
tfidf_dtm_2000,tfidf_vectorizer_2000 = perform_tfidf(CLEANED_TEXT,2000)

# Top 2000 words
top_2000_words = tfidf_vectorizer_2000.get_feature_names()
```

```
The type of tfidf vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
The shape of data matrix  (351237, 2000)
The number of unique words in data matrix  2000
```

In [7]:
```
# Max 5000 features
tfidf_dtm_5000,tfidf_vectorizer_5000 = perform_tfidf(CLEANED_TEXT,5000)

# Top 5000 words
top_5000_words = tfidf_vectorizer_5000.get_feature_names()
```

```
The type of tfidf vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
The shape of data matrix  (351237, 5000)
The number of unique words in data matrix  5000
```

In [8]:
```
# Max 10000 features
tfidf_dtm_10000,tfidf_vectorizer_10000 = perform_tfidf(CLEANED_TEXT,10000)

# Top 10000 words
top_10000_words = tfidf_vectorizer_10000.get_feature_names()
```

```
The type of tfidf vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
The shape of data matrix  (351237, 10000)
The number of unique words in data matrix  10000
```

Let's calculate co-occurrence matrix of separated dataset :

In [9]:
```
# Get Co-occurrence matrix with n-neighbours
def get_co_occurrence_matrix(CLEANED_TEXT,top_words,n_neighbour,co_occurrence_matrix):
    for document in tqdm(CLEANED_TEXT,unit=" document",desc='Co-occurrence matrix'):
        word_list = document.split()
        atleast_two_words = len(set(top_words).intersection(word_list))
        if(atleast_two_words >= 2):
            for index,word in enumerate(word_list):
                if word in top_words:
                    word_to_check = word
                    wi_wj = ()
                    starting_index = max(index-n_neighbour,0)
                    ending_index = min(index+n_neighbour,(len(word_list)-1))
                    for j in range(starting_index,ending_index+1):
                        if word_list[j] in top_words:
                            wi_wj = top_words.index(word_to_check),top_words.index(word_list[j])
                            co_occurrence_matrix[wi_wj[0],wi_wj[1]] += 1
                        else:
                            pass
                else:
                    pass
        else:
            pass

    return co_occurrence_matrix
```

In [10]:
```
# Get co-occurrence matrix for top 2000 features
co_occurrence_matrix_2000 = get_co_occurrence_matrix(CLEANED_TEXT,top_2000_words,5,np.zeros((2000,2000
)))
print("Co-occurrence matrix of first 10 rows X first 10 columns: ")
co_occurrence_matrix_2000[:10,:10]
```

```
Co-occurrence matrix: 100%|██████████████████████████████| 351237/351237 [1:21:44<00:00,
71.61 document/s]
```

```
Co-occurrence matrix of first 10 rows X first 10 columns:
```

Out[10]:
```
array([[1.0794e+04, 6.6000e+01, 8.0000e+00, 2.0000e+00, 9.0000e+00,
        9.0000e+00, 2.9000e+01, 8.0000e+00, 1.2000e+01, 4.0000e+00],
       [6.6000e+01, 1.0979e+04, 8.0000e+00, 9.0000e+00, 4.0000e+00,
        1.5000e+01, 1.5000e+01, 3.0000e+00, 2.1000e+01, 8.0000e+00],
       [8.0000e+00, 8.0000e+00, 1.0140e+03, 0.0000e+00, 3.0000e+00,
        7.0000e+00, 7.0000e+00, 0.0000e+00, 0.0000e+00, 4.0000e+00],
       [2.0000e+00, 9.0000e+00, 0.0000e+00, 7.5700e+02, 1.0000e+00,
        1.0000e+00, 1.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00],
       [9.0000e+00, 4.0000e+00, 3.0000e+00, 1.0000e+00, 1.3360e+03,
        4.0000e+00, 7.0000e+00, 0.0000e+00, 2.0000e+00, 0.0000e+00],
       [9.0000e+00, 1.5000e+01, 7.0000e+00, 1.0000e+00, 4.0000e+00,
        1.7770e+03, 1.4000e+01, 1.0000e+00, 3.0000e+00, 5.0000e+00],
       [2.9000e+01, 1.5000e+01, 7.0000e+00, 1.0000e+00, 7.0000e+00,
        1.4000e+01, 5.8070e+03, 1.0000e+00, 5.0000e+00, 5.0000e+00],
       [8.0000e+00, 3.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
        1.0000e+00, 1.0000e+00, 7.8200e+02, 0.0000e+00, 0.0000e+00],
       [1.2000e+01, 2.1000e+01, 0.0000e+00, 0.0000e+00, 2.0000e+00,
        3.0000e+00, 5.0000e+00, 0.0000e+00, 2.5220e+03, 2.0000e+00],
       [4.0000e+00, 8.0000e+00, 4.0000e+00, 0.0000e+00, 0.0000e+00,
        5.0000e+00, 5.0000e+00, 0.0000e+00, 2.0000e+00, 8.3600e+02]])
```

In [16]:
```python
# Get co-occurrence matrix for top 5000 features
co_occurrence_matrix_5000 = get_co_occurrence_matrix(CLEANED_TEXT,top_5000_words,5,np.zeros((5000,5000
)))
print("Co-occurrence matrix of first 10 rows X first 10 columns: ")
co_occurrence_matrix_2000[:10,:10]
```

```
Co-occurrence matrix: 100%|██████████████████████████████| 351237/351237 [3:52:50<00:00,
25.14 document/s]
```

```
Co-occurrence matrix of first 10 rows X first 10 columns:
```

Out[16]:
```
array([[1.0794e+04, 6.6000e+01, 8.0000e+00, 2.0000e+00, 9.0000e+00,
        9.0000e+00, 2.9000e+01, 8.0000e+00, 1.2000e+01, 4.0000e+00],
       [6.6000e+01, 1.0979e+04, 8.0000e+00, 9.0000e+00, 4.0000e+00,
        1.5000e+01, 1.5000e+01, 3.0000e+00, 2.1000e+01, 8.0000e+00],
       [8.0000e+00, 8.0000e+00, 1.0140e+03, 0.0000e+00, 3.0000e+00,
        7.0000e+00, 7.0000e+00, 0.0000e+00, 0.0000e+00, 4.0000e+00],
       [2.0000e+00, 9.0000e+00, 0.0000e+00, 7.5700e+02, 1.0000e+00,
        1.0000e+00, 1.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00],
       [9.0000e+00, 4.0000e+00, 3.0000e+00, 1.0000e+00, 1.3360e+03,
        4.0000e+00, 7.0000e+00, 0.0000e+00, 2.0000e+00, 0.0000e+00],
       [9.0000e+00, 1.5000e+01, 7.0000e+00, 1.0000e+00, 4.0000e+00,
        1.7770e+03, 1.4000e+01, 1.0000e+00, 3.0000e+00, 5.0000e+00],
       [2.9000e+01, 1.5000e+01, 7.0000e+00, 1.0000e+00, 7.0000e+00,
        1.4000e+01, 5.8070e+03, 1.0000e+00, 5.0000e+00, 5.0000e+00],
       [8.0000e+00, 3.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
        1.0000e+00, 1.0000e+00, 7.8200e+02, 0.0000e+00, 0.0000e+00],
       [1.2000e+01, 2.1000e+01, 0.0000e+00, 0.0000e+00, 2.0000e+00,
        3.0000e+00, 5.0000e+00, 0.0000e+00, 2.5220e+03, 2.0000e+00],
       [4.0000e+00, 8.0000e+00, 4.0000e+00, 0.0000e+00, 0.0000e+00,
        5.0000e+00, 5.0000e+00, 0.0000e+00, 2.0000e+00, 8.3600e+02]])
```

In [17]:
```python
# Get co-occurrence matrix for top 10000 features
co_occurrence_matrix_10000 = get_co_occurrence_matrix(CLEANED_TEXT,top_10000_words,5,np.zeros((10000,1
0000)))
print("Co-occurrence matrix of first 10 rows X first 10 columns: ")
co_occurrence_matrix_2000[:10,:10]
```

```
Co-occurrence matrix: 100%|██████████████████████████████| 351237/351237 [7:57:46<00:00,
12.25 document/s]
```

```
Co-occurrence matrix of first 10 rows X first 10 columns:
```

Out[17]:
```
array([[1.0794e+04, 6.6000e+01, 8.0000e+00, 2.0000e+00, 9.0000e+00,
        9.0000e+00, 2.9000e+01, 8.0000e+00, 1.2000e+01, 4.0000e+00],
       [6.6000e+01, 1.0979e+04, 8.0000e+00, 9.0000e+00, 4.0000e+00,
        1.5000e+01, 1.5000e+01, 3.0000e+00, 2.1000e+01, 8.0000e+00],
       [8.0000e+00, 8.0000e+00, 1.0140e+03, 0.0000e+00, 3.0000e+00,
        7.0000e+00, 7.0000e+00, 0.0000e+00, 0.0000e+00, 4.0000e+00],
       [2.0000e+00, 9.0000e+00, 0.0000e+00, 7.5700e+02, 1.0000e+00,
        1.0000e+00, 1.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00],
       [9.0000e+00, 4.0000e+00, 3.0000e+00, 1.0000e+00, 1.3360e+03,
        4.0000e+00, 7.0000e+00, 0.0000e+00, 2.0000e+00, 0.0000e+00],
       [9.0000e+00, 1.5000e+01, 7.0000e+00, 1.0000e+00, 4.0000e+00,
        1.7770e+03, 1.4000e+01, 1.0000e+00, 3.0000e+00, 5.0000e+00],
       [2.9000e+01, 1.5000e+01, 7.0000e+00, 1.0000e+00, 7.0000e+00,
        1.4000e+01, 5.8070e+03, 1.0000e+00, 5.0000e+00, 5.0000e+00],
       [8.0000e+00, 3.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
        1.0000e+00, 1.0000e+00, 7.8200e+02, 0.0000e+00, 0.0000e+00],
       [1.2000e+01, 2.1000e+01, 0.0000e+00, 0.0000e+00, 2.0000e+00,
        3.0000e+00, 5.0000e+00, 0.0000e+00, 2.5220e+03, 2.0000e+00],
       [4.0000e+00, 8.0000e+00, 4.0000e+00, 0.0000e+00, 0.0000e+00,
        5.0000e+00, 5.0000e+00, 0.0000e+00, 2.0000e+00, 8.3600e+02]])
```

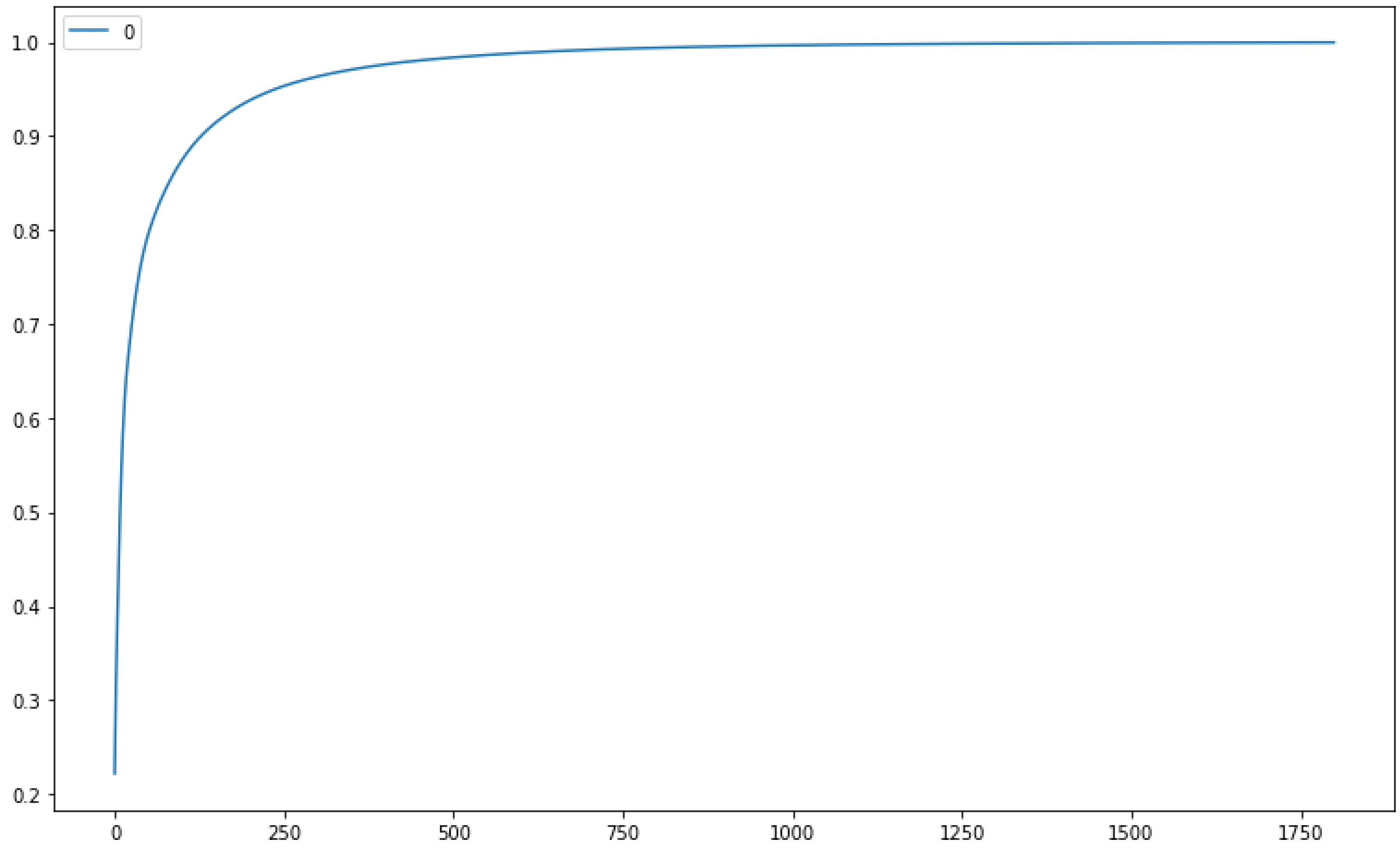
Apply TruncatedSVD on the co-occurrence matrix with optimal n_component value. (calculate n*component(k) using explained* variance_ratio)

In [18]:
```
svd = TruncatedSVD(n_components=1800)
left_singular_vector_2000 = normalize(svd.fit_transform(co_occurrence_matrix_2000), norm="l1")
print(left_singular_vector_2000.shape)
```

```
(2000, 1800)
```

In [19]:
```
print()
pd.DataFrame(np.cumsum(svd.explained_variance_ratio_)).plot(figsize=(13, 8))
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x2e3fd8deb70>



From above plot we can see that around ~250 dimension, we can keep ~95% of the feature information.

In [20]:
```
svd = TruncatedSVD(n_components=250)
left_singular_vector_2000 = normalize(svd.fit_transform(co_occurrence_matrix_2000), norm="l1")
print(left_singular_vector_2000.shape)
```

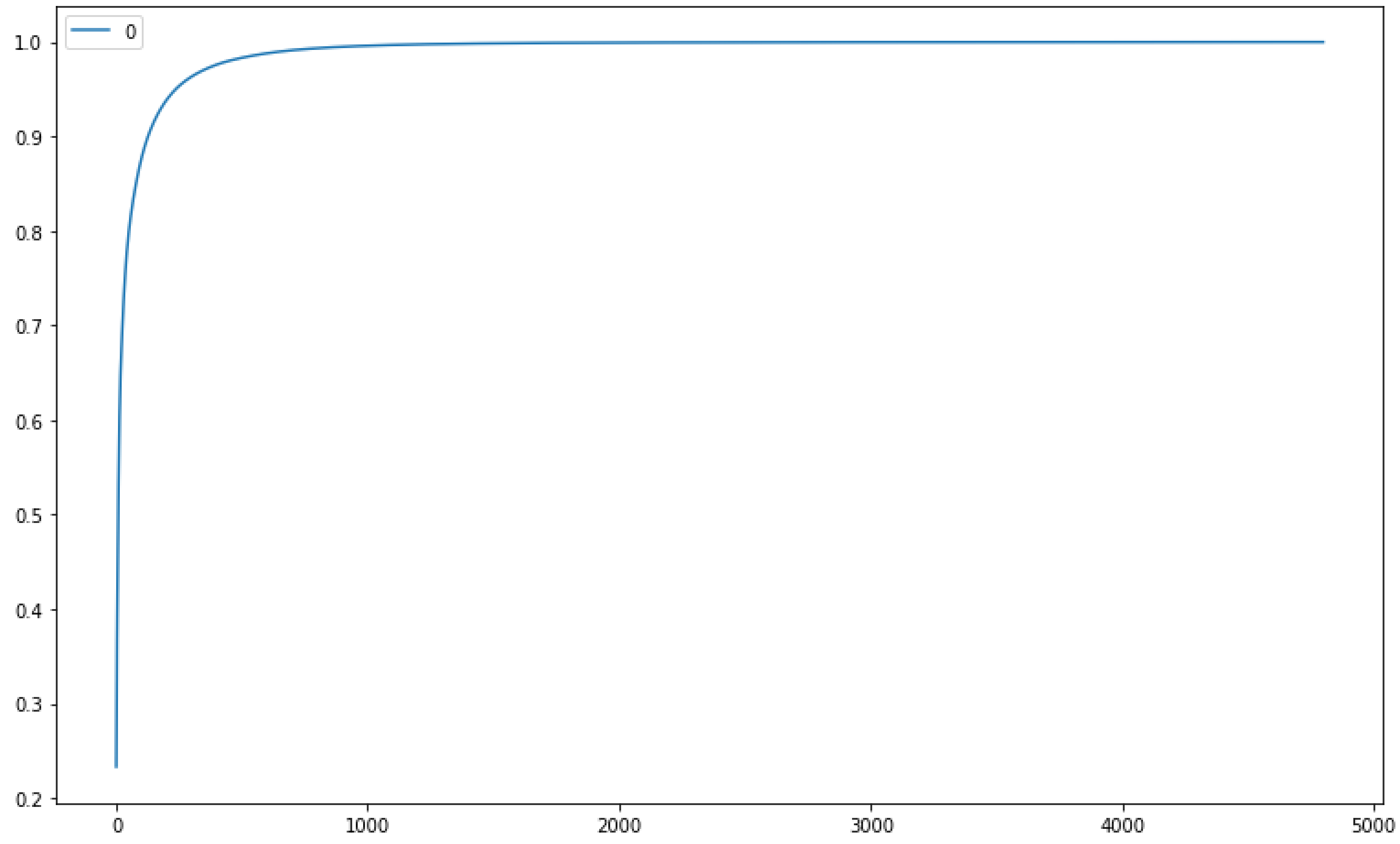
```
(2000, 250)
```

In [21]:
```
print()
svd = TruncatedSVD(n_components=4800)
left_singular_vector_5000 = normalize(svd.fit_transform(co_occurrence_matrix_5000), norm="l1")
print(left_singular_vector_5000.shape)
```

```
(5000, 4800)
```

In [22]:
```
print()
pd.DataFrame(np.cumsum(svd.explained_variance_ratio_)).plot(figsize=(13, 8))
```

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x2e3f7b6d1d0>



From above plot we can see that around ~500 dimension, we can keep ~95% of the feature information.

In [23]:
```
svd = TruncatedSVD(n_components=500)
left_singular_vector_5000 = normalize(svd.fit_transform(co_occurrence_matrix_2000), norm="l1")
print(left_singular_vector_5000.shape)
```

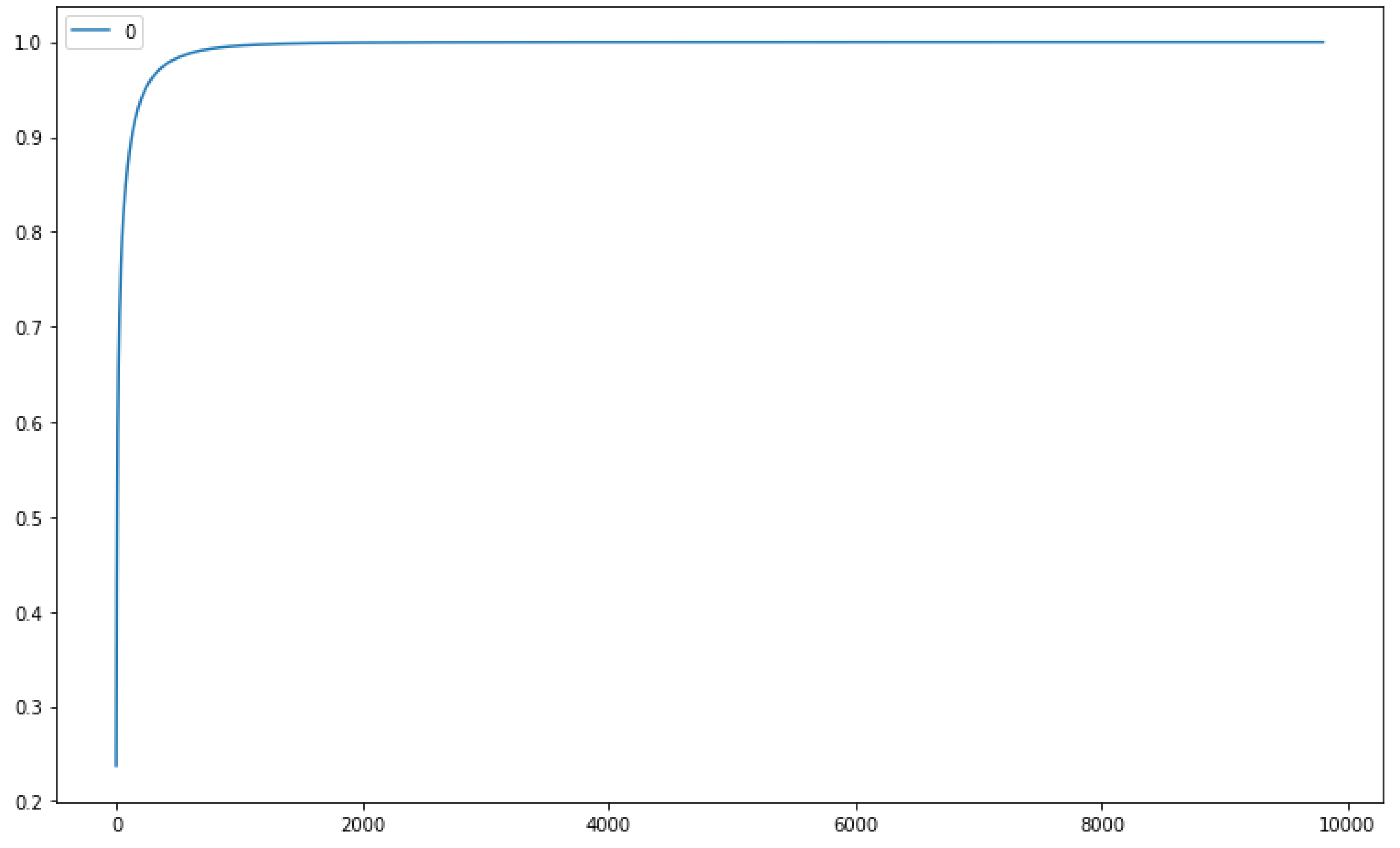
```
(2000, 500)
```

In [24]:
```
print()
svd = TruncatedSVD(n_components=9800)
left_singular_vector_10000 = normalize(svd.fit_transform(co_occurrence_matrix_10000), norm="l1")
print(left_singular_vector_10000.shape)
```

```
(10000, 9800)
```

In [25]:
```
print()
pd.DataFrame(np.cumsum(svd.explained_variance_ratio_)).plot(figsize=(13, 8))
```

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x2e3fb94dac8>



From above plot we can see that around ~1000 dimension, we can keep ~95% of the feature information.

In [26]:

```
svd = TruncatedSVD(n_components=1000)
left_singular_vector_10000 = normalize(svd.fit_transform(co_occurrence_matrix_10000), norm="l1")
print(left_singular_vector_10000.shape)
```

```
(10000, 1000)
```

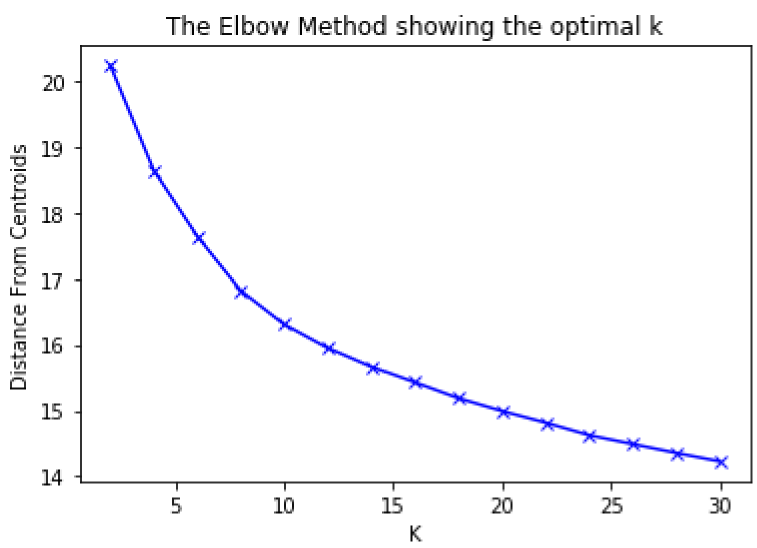
Apply k-means clustering the dataset, and evaluate whether words are semantically similar or not.

In [34]:
```python
# Ideal values of k-clusters
k_clusters = [2,4,6,8,10,12,14,16,18,20,22,24,26,28,30]

dist_from_centroids = []

def perform_kmeans(X):
    dist_from_centroids = []

    for k_value in tqdm(k_clusters,unit=" k-values",desc='Perform K-Means'):
        # Instantiate KMeans
        kmeans = KMeans(n_clusters= k_value,
                        init="k-means++",
                        n_init=100,
                        max_iter=5000,
                        n_jobs=-1)

        # Fit model to data
        kmeans.fit(X)

        # Save data for plotting
        dist_from_centroids.append(kmeans.inertia_)

    return k_clusters,dist_from_centroids

def get_clusters(kmeans, top_words):
    index = [i for i in range(len(top_words))]

    cluster_label_word_indices = dict()
    for cluster_label, index in zip(kmeans.labels_, index) :
        cluster_label_word_indices.setdefault(cluster_label,[])
        cluster_label_word_indices[cluster_label].append(index)


    clusters = dict()
    cluster_labels_with_no_duplicate = sorted(list(set(kmeans.labels_)))
    for i in cluster_labels_with_no_duplicate:
        holder = []
        list_word_indices = cluster_label_word_indices[i]
        for word_index in sorted(list_word_indices):
            holder.append(top_words[word_index])
        clusters[i] = holder


    return clusters


def plot_word_cloud(words,top_words,frequencies):
    wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = set(STOPWORDS),
                min_font_size = 10)

    wordcloud.generate_from_frequencies({word : frequencies[top_words.index(word)] for word in words})

    # plot the WordCloud image
    plt.figure(figsize = (8, 8), facecolor = None)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.tight_layout(pad = 0)
    plt.show()


def plot_elbow(k_clusters,dist_from_centroids):
    print()
    print()
    # plot Sum of squared distances of samples to their closest cluster center vs 'K' value

    # Plot the elbow
    plt.plot(np.array(k_clusters), np.array(dist_from_centroids), 'bx-')
    plt.xlabel('K')
    plt.ylabel('Distance From Centroids')
    plt.title('The Elbow Method showing the optimal k')
    plt.show()

def print_intra_cluster_words(clusters,word_count=25):
    ptable=PrettyTable()
    ptable.title = "*** {0} Most Common Clustered Words ***".format(word_count)
    ptable.field_names=["Cluster Number","Words"]
```

```python
    for key, value in clusters.items():
        temp = []
        newline = 1
        value = value[0:word_count]
        for word in value:
            temp.append(word)
            if newline%10 == 0:
                temp.append('\n')
            newline += 1

        ptable.add_row([str(key),','.join(temp)])
        ptable.add_row(['\n','\n'])

    print(ptable)

def plot_clusters(kmeans,data_matrix_2_dim,label_color_map):
    # Compute cluster centers and predict cluster indices
    cluster_labels = kmeans.fit_predict(data_matrix_2_dim)

    # Define our own color map
    label_color = [label_color_map[l] for l in cluster_labels]

    # Plot the scatter digram
    plt.figure(figsize = (12,10))
    for index in range(0,len(label_color)) :
        label_name = list(label_color_map.keys())[list(label_color_map.values()).index(label_color[ind
ex])]
        plt.scatter(data_matrix_2_dim[index,0],data_matrix_2_dim[index,1], c= label_color[index],label
=label_name, alpha=0.8)
    plt.xlabel("Component 1")
    plt.ylabel("Component 2")
    handles, labels = plt.gca().get_legend_handles_labels()
    by_label = OrderedDict(zip(labels, handles))
    plt.legend(by_label.values(), by_label.keys())
    plt.show()
```

In [28]:
```python
# Perfrom K-Means
k_clusters,dist_from_centroids = perform_kmeans(left_singular_vector_2000)

# Plot elbow-knee figure and evaluate best K
plot_elbow(k_clusters,dist_from_centroids)
```

```
Perform K-Means: 100%|██████████████████████████████| 15/15 [02:01<00:00,
10.35s/ k-values]
```



After ~7-8, steepness of the curve starting to reduce, so we will take K = 8.

In [29]:
```python
# Instantiate k-means with optimal K
kmeans = KMeans(n_clusters= 8,
                        init="k-means++",
                        n_init=100,
                        max_iter=5000,
                        n_jobs=-1)
```

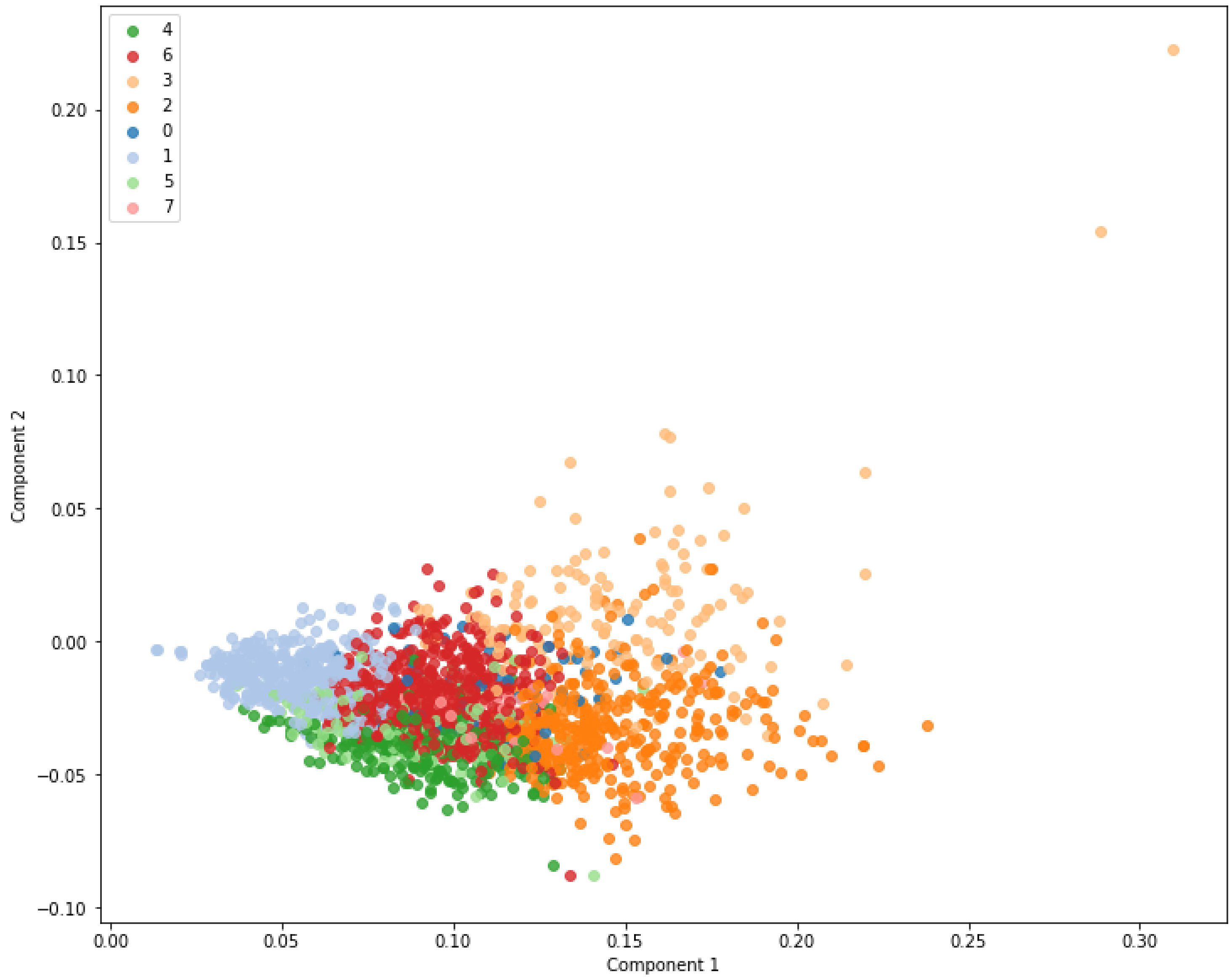
In [30]:
```python
 # Fit model to data
kmeans = kmeans.fit(left_singular_vector_2000)
kmeans
```

Out[30]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=5000,
    n_clusters=8, n_init=100, n_jobs=-1, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [37]:
```python
# get clusters in the form of dictionary - (key,value) => (cluster_label,list_of_words)
clusters = get_clusters(kmeans,top_2000_words)

# Print words in each cluster
print_intra_cluster_words(clusters,word_count=50)
```

```
+-----------------------------------------------------------------------------------------------+
|                                *** 50 Most Common Clustered Words ***                          |
+----------------+------------------------------------------------------------------------------+
| Cluster Number |                                     Words                                    |
+----------------+------------------------------------------------------------------------------+
|       0        |    bigelow,british,calm,celesti,chai,chamomil,chines,earl,english,flower,    |
|                |    ,fragrant,green,grey,herbal,india,infus,irish,japan,japanes,jasmin,       |
|                |    ,leaf,lipton,loos,matcha,numi,oolong,peach,pearl,peppermint,pitcher,      |
|                |    ,relax,rooibo,rose,sooth,stash,steep,tazo,tea,teabag,tip,                 |
|                |    ,twine,yogi                                                               |
|                |                                                                              |
|                |                                                                              |
|       1        |    activ,ador,adult,age,allerg,allergi,anim,appetit,ate,attent,              |
|                |    ,babi,bakeri,ball,beg,began,biscuit,blue,bone,breath,breed,               |
|                |    ,brush,buffalo,bulli,busi,cancer,canin,cat,caus,challeng,chef,            |
|                |    ,chewer,choke,coat,comfort,crazi,danger,daughter,dental,destroy,develop,  |
|                |    ,devour,diagnos,diamond,diarrhea,die,digest,diseas,dog,duck,ear,          |
|                |                                                                              |
|                |                                                                              |
|       2        |    abl,advertis,afford,amazon,anymor,anywher,area,asian,ask,auto,            |
|                |    ,automat,bargain,becam,buck,bulk,buy,buyer,california,canada,cancel,      |
|                |    ,canist,carri,carton,chain,charg,cheaper,cheapest,check,citi,club,        |
|                | ,com,competit,consider,contact,continu,conveni,correct,costco,couldnt,coupon, |
|                | ,current,custom,deal,decid,deliv,deliveri,descript,difficult,direct,disappear, |
|                |                                                                              |
|                |                                                                              |
|       3        |    absolut,absorb,adjust,afternoon,afterward,agav,air,alcohol,also,altern,   |
|                |    ,amaz,anni,antioxid,anytim,apart,appl,appli,approxim,aspartam,balanc,     |
|                |    ,balsam,banana,base,basil,batch,batter,bbq,beauti,bed,beef,               |
|                |    ,behind,bell,belli,better,bigger,bite,blender,blow,bodi,bonus,            |
|                |    ,boost,bother,bottom,bowl,boyfriend,bran,break,bright,bring,broccoli,     |
|                |                                                                              |
|                |                                                                              |
|       4        |    accept,accord,act,addict,addit,admit,advis,affect,afraid,agre,            |
|                |    ,ahead,aid,allow,alon,along,alot,alreadi,although,america,american,       |
|                |    ,among,answer,anyway,appar,appear,appreci,arent,asid,associ,assort,       |
|                |    ,assum,attempt,attract,averag,avoid,awar,awesom,awhil,basic,bear,         |
|                |    ,becom,begin,believ,besid,bet,bewar,beyond,biggest,bill,bore,             |
|                |                                                                              |
|                |                                                                              |
|       5        |    across,actual,ad,add,ago,almond,almost,alway,amount,anoth,                |
|                |    ,anyon,anyth,around,arriv,avail,away,back,bad,bag,bake,                   |
|                |    ,bar,barley,basi,basket,bean,beat,benefit,best,big,birthday,              |
|                |    ,bit,bitter,black,blend,blood,bob,boil,bonsai,bottl,bought,               |
|                |    ,box,bpa,brand,bread,breakfast,breast,brew,broken,brought,brown,          |
|                |                                                                              |
|                |                                                                              |
|       6        |    acai,acquir,aftertast,appeal,apricot,aroma,aromat,artifici,authent,aw,    |
|                |    ,bacon,bare,beer,berri,beverag,blackberri,bland,blueberri,bud,burn,       |
|                |  ,burnt,burst,butteri,caramel,cardboard,chemic,cinnamon,citrus,classic,clove, |
|                |    ,cola,combin,combo,complex,compliment,consist,cough,creami,crisp,deep,    |
|                |    ,delic,describ,detect,dilut,disgust,dislik,distinct,earthi,enhanc,fake,   |
|                |                                                                              |
|                |                                                                              |
|       7        |    acid,bay,bold,brewer,cafe,caffein,cappuccino,coffe,creamer,crema,         |
|                |    ,cup,darker,decaf,donut,drank,drinker,drip,eight,espresso,expert,         |
|                |    ,filter,folger,fuel,grind,grinder,ground,hazelnut,illi,instant,intern,    |
|                |    ,island,kcup,keurig,kona,latt,lavazza,lover,machin,magic,maker,           |
|                |    ,mate,maxwel,medium,melitta,mocha,mountain,mug,pod,pot,press,             |
|                |                                                                              |
|                |                                                                              |
|                |                                                                              |
+----------------+------------------------------------------------------------------------------+
```

In [38]:
```python
# Color map
cluster_colors = {0: '#1f77b4', 1: '#aec7e8', 2: '#ff7f0e', 3: '#ffbb78', 4: '#2ca02c',
                  5: '#98df8a', 6: '#d62728', 7: '#ff9896'}


# Plot clusters
plot_clusters(kmeans,left_singular_vector_2000,cluster_colors)
```

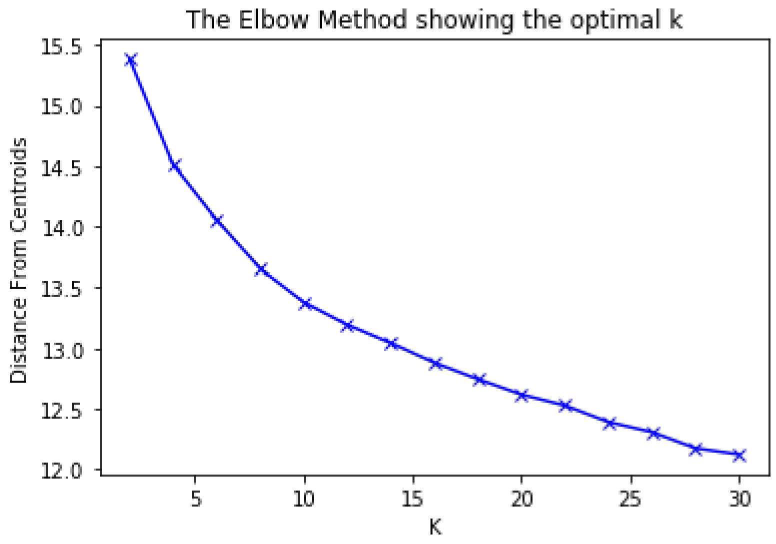


In [39]:
```
print("Words in cluster {0} :\n".format(clusters[0]))
print()
print()
plot_word_cloud(clusters[0],top_2000_words,tfidf_vectorizer_2000.idf_)
```

```
Words in cluster ['bigelow', 'british', 'calm', 'celesti', 'chai', 'chamomil', 'chines', 'earl', 'engl
ish', 'flower', 'fragrant', 'green', 'grey', 'herbal', 'india', 'infus', 'irish', 'japan', 'japanes',
'jasmin', 'leaf', 'lipton', 'loos', 'matcha', 'numi', 'oolong', 'peach', 'pearl', 'peppermint', 'pitch
er', 'relax', 'rooibo', 'rose', 'sooth', 'stash', 'steep', 'tazo', 'tea', 'teabag', 'tip', 'twine', 'y
ogi'] :
```

In [40]:
```python
# Perfrom K-Means
k_clusters,dist_from_centroids = perform_kmeans(left_singular_vector_5000)

# Plot elbow-knee figure and evaluate best K
plot_elbow(k_clusters,dist_from_centroids)
```

```
Perform K-Means: 100%|██████████████████████████████████████████████████| 15/15 [04:04<00:00,
20.67s/ k-values]
```



After ~8-10, steepness of the curve starting to reduce, so we will take K = 9.

In [41]:
```python
# Instantiate k-means with optimal K
kmeans = KMeans(n_clusters= 9,
                        init="k-means++",
                        n_init=100,
                        max_iter=5000,
                        n_jobs=-1)
```

In [42]:
```python
 # Fit model to data
kmeans = kmeans.fit(left_singular_vector_5000)
kmeans
```

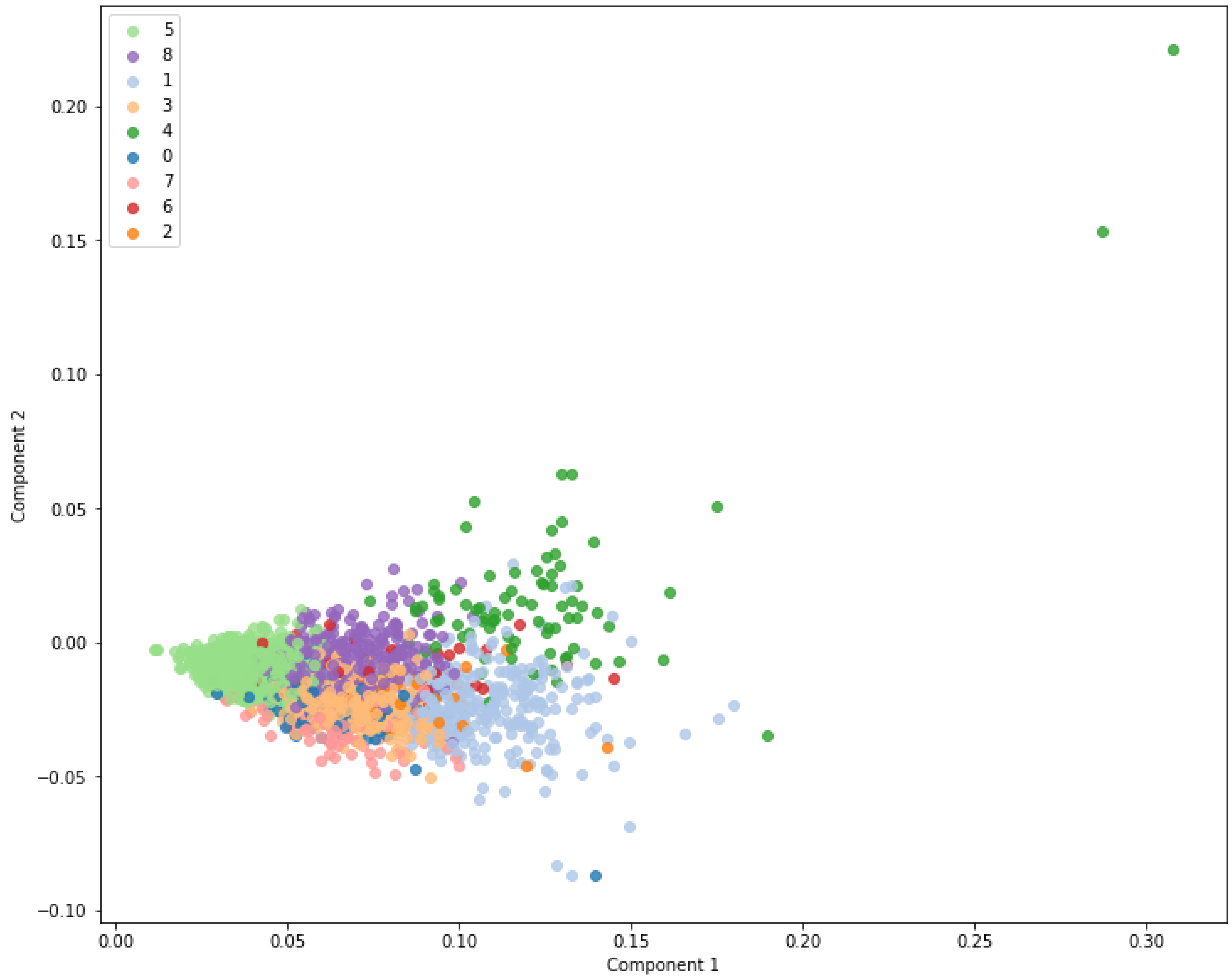
Out[42]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=5000,
    n_clusters=9, n_init=100, n_jobs=-1, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

In [44]:
```python
# get clusters in the form of dictionary - (key,value) => (cluster_label,list_of_words)
clusters = get_clusters(kmeans,top_5000_words)

# Print words in each cluster
print_intra_cluster_words(clusters,word_count=50)
```

*** 50 Most Common Clustered Words ***

Cluster Number	Words
0	access,accustom,ach,ad,admit,afraid,altogeth,altura,angri,anyway, ,appropri,art,arthriti,ass,averag,backyard,balm,barbecu,barilla,basmati, ,belong,blair,bodi,bouquet,bpa,brach,bragg,branch,breadmak,bright, ,bucket,budget,bulldog,bulli,cane,capsul,car,caramel,carb,caribbean, ,carolina,cat,caveat,charg,cheer,chef,chewi,chocol,clarifi,cleaner,
1	absolut,absorb,accid,accident,accompani,action,actual,adequ,africa,aftertast, ,airtight,alert,along,altoid,aluminum,alway,americano,amish,amongst,angl, ,answer,antibiot,antioxid,antler,anywher,appetit,applesauc,appli,approx,articl, ,artisan,asham,ask,assur,asthma,astring,attempt,awar,awesom,babi, ,backpack,bait,bar,barbequ,barri,base,basic,basil,batch,batter,
2	abund,act,activ,addict,address,adopt,advantag,advers,afternoon,agent, ,agre,aid,air,airport,alittl,alkalin,allerg,alot,altern,amus, ,analysi,ancient,anim,anni,annual,anymor,anyth,argu,arm,around, ,asap,ascorb,asia,asian,assam,assembl,ate,attack,aussi,authent, ,autoship,avid,avoid,await,awak,back,bacon,bacteria,bad,bam,
3	among,approach,arthur,artichok,attend,aw,began,bland,blech,blond, ,bold,brisket,brother,brule,brush,brussel,bundl,calcul,caribou,cauliflow, ,chanc,choke,cholesterol,clementin,combo,compound,concept,constant,contain,cool, ,cornmeal,cornstarch,crispi,crunchi,cur,cure,cvs,dear,deck,deeper, ,drastic,ethic,ethiopian,exercis,fatti,five,flow,flu,frank,gain,
4	abil,abl,abus,accent,accomplish,accord,adapt,adjust,ador,adult, ,advertis,advic,advis,aerat,aficionado,african,agav,age,aint,akin, ,alarm,alaska,albacor,ale,alon,alreadi,alright,alter,although,amaranth, ,amazon,amber,america,ami,amino,amoretti,amount,ampl,anchovi,angel, ,anniversari,annoy,anoth,anybodi,anytim,apart,apolog,appar,appet,appl,
5	anyhow,artifici,bag,balsam,basement,buffalo,burst,cent,chocohol,chocolati, ,clove,competit,con,confess,confirm,costa,cramp,crema,cuppa,deploy, ,dew,discontinu,discov,dish,eleven,entir,environ,fell,flake,flavorless, ,freezer,french,frequent,gambl,gloria,hairbal
6	absurd,accept,account,accur,achiev,acid,acn,across,add,addit, ,advanc,adventur,advert,affect,afford,afghanistan,aggress,ahead,ahoy,airi, ,aka,alcohol,alfalfa,alfredo,alik,aliv,allow,almond,alo,american, ,anis,ant,anticip,anxieti,anxious,anyon,apricot,april,arizona,assum, ,banana,barney,barrel,bat,bath,beagl,bear,beat,becom,behind,
7	acai,ahmad,ala,allergen,almost,also,amaretto,appeal,appear,aspect, ,astound,atkin,awhil,beach,begun,berger,bother,bout,brace,breadstick, ,breez,brewer,brick,buildup,bush,canadian,candi,casserol,celtic,chamomil, ,charcoal,chore,clay,coat,coffeemak,compon,controversi,copper,counterpart,cover, ,cow,cracker,crusti,cute,cycl,dal,dandruff,david,definit,delicaci,
8	acquir,aerogarden,afterward,ago,aisl,albeit,allergi,allevi,allot,amaz, ,aspartam,assist,atlanta,attach,august,auto,automat,baggi,ball,ban, ,band,bang,bbq,bedtim,belgium,benefit,bewar,biscotti,bittersweet,blossom, ,bottom,breadcrumb,breastmilk,breath,brilliant,broccoli,btw,bum,buri,cacao, ,caffien,cap,carnat,catch,celiac,cellophan,charger,chicori,choos,chop,

In [45]:
```python
# Color map
cluster_colors = {0: '#1f77b4', 1: '#aec7e8', 2: '#ff7f0e', 3: '#ffbb78', 4: '#2ca02c',
                  5: '#98df8a', 6: '#d62728', 7: '#ff9896', 8: '#9467bd'}

# Plot clusters
plot_clusters(kmeans,left_singular_vector_5000,cluster_colors)
```

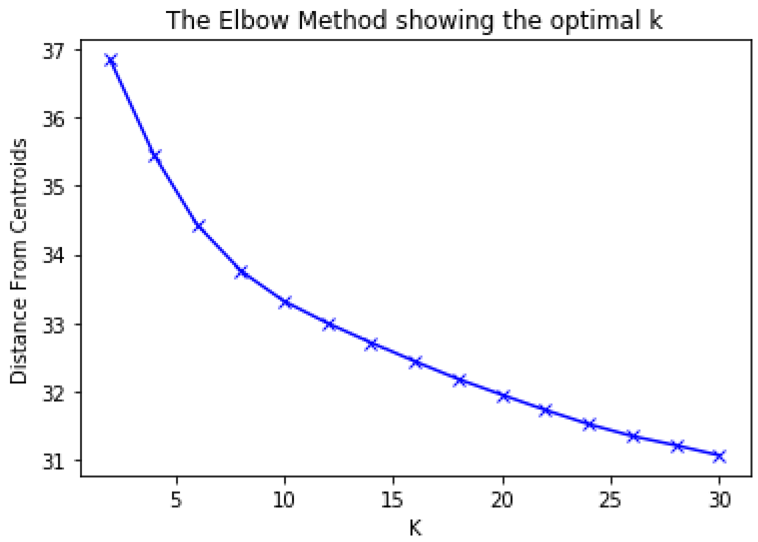


In [56]:
```
print("Words in cluster {0} :\n".format(clusters[8]))
print()
print()
plot_word_cloud(clusters[8],top_5000_words,tfidf_vectorizer_5000.idf_)
```

```
Words in cluster ['acquir', 'aerogarden', 'afterward', 'ago', 'aisl', 'albeit', 'allergi', 'allevi',
'allot', 'amaz', 'aspartam', 'assist', 'atlanta', 'attach', 'august', 'auto', 'automat', 'baggi', 'bal
l', 'ban', 'band', 'bang', 'bbq', 'bedtim', 'belgium', 'benefit', 'bewar', 'biscotti', 'bittersweet',
'blossom', 'bottom', 'breadcrumb', 'breastmilk', 'breath', 'brilliant', 'broccoli', 'btw', 'bum', 'bur
i', 'cacao', 'caffien', 'cap', 'carnat', 'catch', 'celiac', 'cellophan', 'charger', 'chicori', 'choo
s', 'chop', 'classic', 'cleans', 'clearanc', 'coars', 'colon', 'comment', 'compress', 'consid', 'cream
i', 'creamier', 'critic', 'crock', 'crocker', 'dane', 'devil', 'diamond', 'digest', 'dimens', 'diminis
h', 'dream', 'dress', 'drive', 'dull', 'dutch', 'earth', 'endors', 'england', 'essenc', 'estim', 'exam
pl', 'exclus', 'expir', 'exposur', 'exquisit', 'fact', 'factor', 'fall', 'fear', 'field', 'flame', 'fl
oor', 'focus', 'folk', 'forev', 'forget', 'forgiv', 'franc', 'freez', 'front', 'fur', 'fuss', 'genet',
'georgia', 'glue', 'googl', 'gosh', 'grain', 'grand', 'grapefruit', 'grass'] :
```


```
In [47]: # Perfrom K-Means
k_clusters,dist_from_centroids = perform_kmeans(left_singular_vector_10000)

# Plot elbow-knee figure and evaluate best K
plot_elbow(k_clusters,dist_from_centroids)
```

```
Perform K-Means: 100%|██████████████████████████████████████████████████| 15/15 [1:06:49<00:00, 3
57.55s/ k-values]
```



After ~8-10, steepness of the curve starting to reduce, so we will take K = 10.

```
In [57]: # Instantiate k-means with optimal K
kmeans = KMeans(n_clusters= 10,
                init="k-means++",
                n_init=100,
                max_iter=5000,
                n_jobs=-1)
```

```
In [58]: # Fit model to data
kmeans = kmeans.fit(left_singular_vector_10000)
kmeans
```

```
Out[58]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=5000,
    n_clusters=10, n_init=100, n_jobs=-1, precompute_distances='auto',
    random_state=None, tol=0.0001, verbose=0)
```

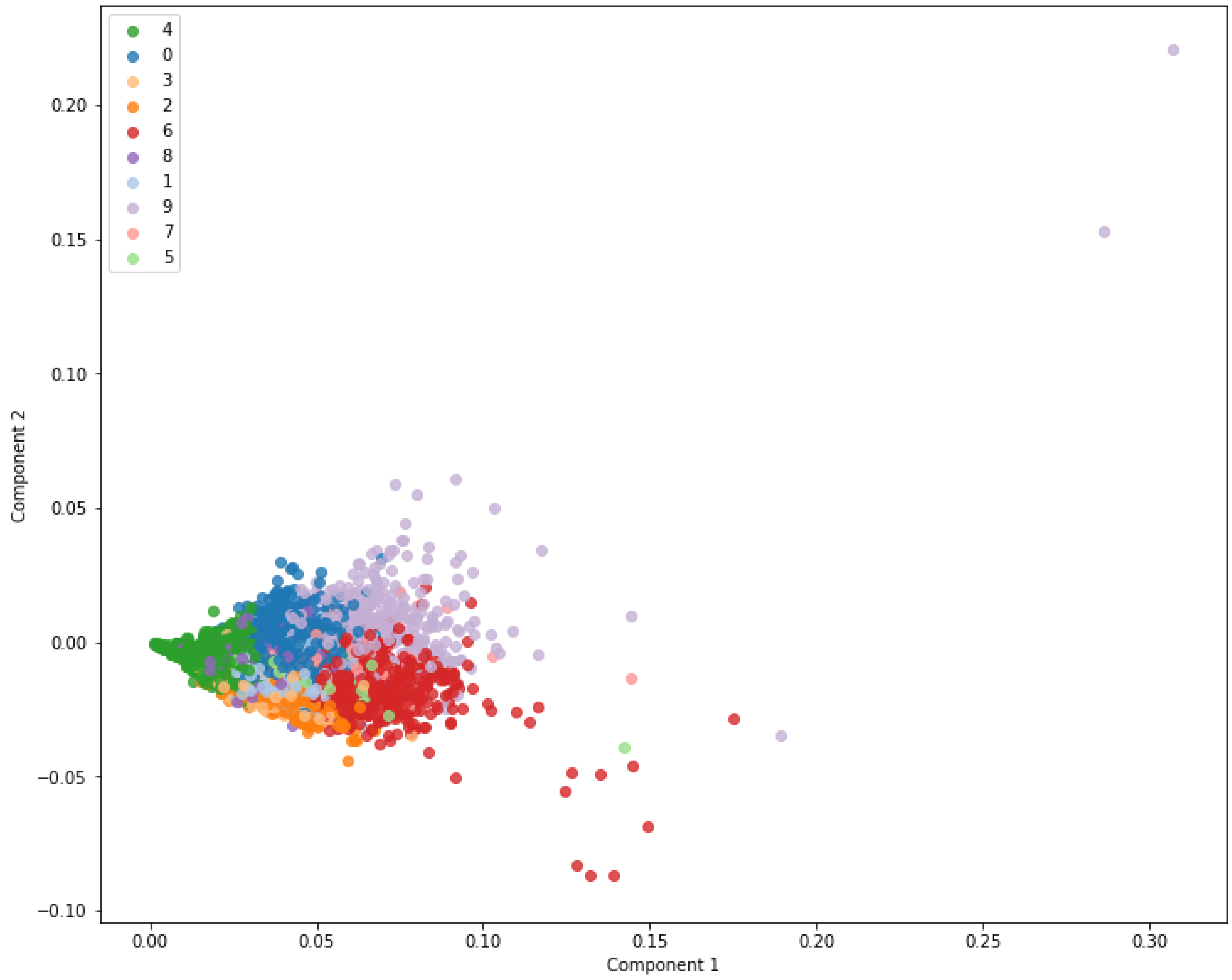
In [59]:
```
# get clusters in the form of dictionary - (key,value) => (cluster_label,list_of_words)
clusters = get_clusters(kmeans,top_10000_words)

# Print words in each cluster
print_intra_cluster_words(clusters,word_count=50)
```

*** 50 Most Common Clustered Words ***

Cluster Number	Words
0	abc,abl,abroad,absurd,account,accross,acknowledg,acm,address,advert, ,affili,afford,afghanistan,againi,agreement,airport,aisl,alabama,alaska,albertson, ,aldi,alpen,amazom,amazon,amazoni,amz,amzn,angri,ann,antonio, ,anymor,anyplac,anywher,apiec,apo,apolog,appal,approx,april,area, ,ariv,arkansa,arrang,arriv,asap,asham,asian,assumpt,astronom,atlanta,
1	aaa,aafco,abbey,abdomin,abnorm,abound,absinth,absorpt,abv,acacia, ,academi,acceler,accomod,accumul,ace,acerola,acesulfam,acet,acid,acidophilus, ,acn,across,actor,ade,adhd,adher,adject,administ,admiss,adren, ,adult,adulthood,advertis,advoc,aerat,aerogarden,aerogrow,afterlif,afternoon,agar, ,agav,age,agenc,agenda,ago,agoni,agricultur,ahem,ahi,ahoy,
2	aback,abomin,absenc,absent,absolut,absolutley,acclim,accuraci,adam,addict, ,addit,admir,admit,adulter,aero,aesthet,affair,affin,aforement,afteral, ,aftertast,agre,agreeabl,ahh,aid,airhead,airi,albanes,ale,alien, ,alik,alley,alli,allur,almond,alo,alon,alpin,altern,altho, ,although,altoid,amateur,amaz,amber,ambrosia,ami,ammonia,amo,amp,
3	abandon,acana,action,activ,administr,adopt,advisor,aggress,agil,agress, ,airedal,akc,akita,alergi,alpha,alpo,anal,analysi,antler,apso, ,artemi,arthriti,aussi,australian,avoderm,bakeri,bank,barf,basset,beagl, ,beg,beggin,behavior,bella,belov,bench,bene,bengal,bernard,bff, ,bichon,bil,bison,bistro,bladder,blockag,bone,bonker,booda,border,
4	abt,abus,accessori,accommod,accord,accus,ach,acquaint,acr,acut, ,acv,adapt,adhes,advanc,advers,advil,aggrav,agit,ahhh,ahhhh, ,ahold,aim,air,airlin,airplan,airtight,alarm,alert,aliv,alleg, ,allevi,allow,alreadi,aluminum,amount,ampl,amus,andi,angl,ankl, ,announc,anoth,ant,antacid,anti,antibiot,antiqu,anxieti,appear,appli,

```
        |
|               |
        |
|       5       |  acidi,aeropress,afficionado,aficionado,aloha,altura,americano,arabica,backcountri,
bali,   |
|               |             ,barista,baronet,bay,bean,berr,bialetti,blanc,blend,bodum,bold,
        |
|               |        ,bolder,boldest,bosch,boyer,brazilian,brevill,brew,brewer,britt,brooklyn,
        |
|               |               ,bros,brule,bunn,burr,bustelo,caf,cafe,caff,caffien,caffin,
        |
|               | ,cafix,camano,cameron,cappaccino,cappuccino,cappuchino,cappucino,capresso,capuccin
o,caraf, |
|               |
        |
|               |
        |
|               |
        |
|       6       |           adagio,africa,african,ahmad,antioxid,arizona,assam,astring,aveda,avid,
        |
|               |      ,barri,bedtim,bentley,bergamont,bergamot,bewley,bigelow,black,blackcurr,blosso
m,      |
|               |             ,boba,brisk,brit,british,burk,bush,caddi,calm,camellia,camomil,
        |
|               |      ,caravan,cardamon,catechin,celesti,ceremoni,ceylon,cha,chakra,chamomil,chante
a,       |
|               |         ,chest,chi,chines,chrysanthemum,coca,cozi,cuppa,dalfour,dandelion,darjeel,
        |
|               |
        |
|               |
        |
|               |
        |
|       7       |                 absorb,ad,add,adobo,agent,aji,alessi,alfredo,alongsid,amd,
        |
|               |    ,ancho,anchovi,ancient,andouill,antipasto,arborio,arrowroot,artichok,arugula,asi
ago,    |
|               |           ,asparagus,atop,avacado,avocado,baba,badia,baguett,baja,bake,balsam,
        |
|               |          ,barbecu,barbequ,barilla,barley,baron,bartend,base,basil,basmati,bast,
        |
|               |               ,batter,bbq,bbqs,beaten,beater,beef,beliz,bell,bertolli,betti,
        |
|               |
        |
|               |
        |
|               |
        |
|       8       |          acai,accent,accentu,accustom,achiev,acquir,acrid,akin,alcohol,alright,
        |
|               |          ,alter,amaretto,anis,appeal,appletini,apricot,aquir,aromat,artif,ash,
        |
|               |          ,ashtray,authent,awe,background,bare,beani,bearabl,bit,bitter,bizarr,
        |
|               |       ,blackberri,blah,blander,blech,bubblegum,bud,burnt,burst,butteri,butterscotc
h,       |
|               |    ,captur,cardboard,carmel,chalk,chalki,charact,characterist,cheesi,cheeto,chocola
tey,    |
|               |
        |
|               |
        |
|               |
        |
|       9       |         abil,abund,accept,access,accid,accident,accompani,accomplish,accur,act,
        |
|               |          ,actual,adequ,adjust,ador,advantag,adventur,advic,advis,affect,afraid,
        |
|               |              ,afterward,ahead,aint,aka,ala,albeit,alittl,allot,almost,alot,
        |
|               |          ,also,altogeth,alway,america,among,amongst,anni,annoy,answer,anticip,
        |
|               |       ,anxious,anybodi,anyhow,anytim,anyway,appar,appet,appetit,applesauc,applic,
        |
|               |
        |
|               |
        |
|               |
        |
+----------------+------------------------------------------------------------------------------------
--------+
```

In [60]:
```
# Color map
cluster_colors = {0: '#1f77b4', 1: '#aec7e8', 2: '#ff7f0e', 3: '#ffbb78', 4: '#2ca02c',
                  5: '#98df8a', 6: '#d62728', 7: '#ff9896', 8: '#9467bd', 9: '#c5b0d5'}

# Plot clusters
plot_clusters(kmeans,left_singular_vector_10000,cluster_colors)
```



In [61]:
```
print("Words in cluster {0} :\n".format(clusters[0]))
print()
print()
plot_word_cloud(clusters[0],top_10000_words,tfidf_vectorizer_10000.idf_)
```

```
Words in cluster ['abc', 'abl', 'abroad', 'absurd', 'account', 'accross', 'acknowledg', 'acm', 'addres
s', 'advert', 'affili', 'afford', 'afghanistan', 'againi', 'agreement', 'airport', 'aisl', 'alabama',
'alaska', 'albertson', 'aldi', 'alpen', 'amazom', 'amazon', 'amazoni', 'amz', 'amzn', 'angri', 'ann',
'antonio', 'anymor', 'anyplac', 'anywher', 'apiec', 'apo', 'apolog', 'appal', 'approx', 'april', 'are
a', 'ariv', 'arkansa', 'arrang', 'arriv', 'asap', 'asham', 'asian', 'assumpt', 'astronom', 'atlanta',
'aug', 'august', 'austin', 'australia', 'author', 'auto', 'autom', 'automat', 'autoship', 'avail', 'av
enu', 'await', 'backord', 'balk', 'bargin', 'basi', 'behold', 'beleiv', 'beliv', 'berkeley', 'bevmo',
'bingo', 'bjs', 'bolivia', 'boo', 'bought', 'boutiqu', 'box', 'brainer', 'brows', 'buck', 'bulk', 'bun
dl', 'butcher', 'calcul', 'calib', 'calif', 'california', 'came', 'camper', 'canada', 'cancel', 'car
d', 'cari', 'carol', 'carolina', 'carri', 'cart', 'case', 'cash', 'catalog', 'cbd', 'ceas', 'cent', 'c
entral', 'chain', 'charg', 'chariti', 'cheaper', 'cheapest', 'cheapli', 'check', 'checkout', 'cheepe
r', 'chicago', 'chinatown', 'cite', 'citi', 'clearanc', 'clerk', 'click', 'club', 'cmon', 'coast', 'co
de', 'colorado', 'columbus', 'com', 'commend', 'commissari', 'commod', 'communic', 'compani', 'competi
t', 'comput', 'confirm', 'connecticut', 'contempl', 'continu', 'coop', 'copi', 'cornnut', 'corpor', 'c
orrect', 'cosco', 'cost', 'costco', 'couldnt', 'counti', 'countri', 'coupon', 'courtesi', 'credit', 'c
rimin', 'cruis', 'cub', 'current', 'cvs', 'dalla', 'deal', 'dealer', 'dealt', 'dear', 'dec', 'decemb',
'decept', 'decid', 'dedic', 'defect', 'definat', 'definitley', 'delawar', 'delay', 'delet', 'deli', 'd
eliv', 'deliveri', 'denver', 'deploy', 'depot', 'dept', 'destin', 'devast', 'diego', 'difficult', 'dis
abl', 'disclos', 'discontinu', 'discount', 'discrep', 'discript', 'dismay', 'display', 'displeas', 'di
ssatisfact', 'dissatisfi', 'distanc', 'distributor', 'dixi', 'dollar', 'doorstep', 'dot', 'downtown',
'drive', 'drove', 'drug', 'drugstor', 'dupe', 'durke', 'earlier', 'east', 'eastern', 'ebay', 'econom
i', 'ecstat', 'effici', 'elat', 'elig', 'elsewher', 'email', 'employe', 'england', 'error', 'escal',
'estim', 'ethic', 'ethnic', 'euro', 'everywher', 'exchang', 'excit', 'exhorbit', 'exorbit', 'expedi',
'expedit', 'expens', 'explan', 'export', 'facebook', 'farmer', 'fast', 'fastest', 'fault', 'faulti',
'feb', 'februari', 'fedex', 'fee', 'feedback', 'fifti', 'file', 'final', 'financi', 'find', 'florida',
'florist', 'fluctuat', 'foodstuff', 'foolish', 'found', 'fraction', 'fraud', 'fred', 'free', 'freebi',
'freight', 'friday', 'frontier', 'fulfil', 'futur', 'gambl', 'georgia', 'germani', 'glad', 'glitch',
'global', 'gnc', 'googl', 'got', 'goug', 'gracious', 'grandaught', 'greedi', 'grocer', 'groceri', 'gro
ssli', 'halal', 'happi', 'hardwar', 'harri', 'hassl', 'haul', 'healthfood', 'heartbeat', 'heb', 'heft
i', 'hero', 'highway', 'hispan', 'histori', 'hometown', 'honor', 'hoo', 'hooray', 'hope', 'houston',
'hurray', 'idaho', 'iherb', 'ikea', 'illinoi', 'imag', 'imposs', 'improp', 'impuls', 'inc', 'incom',
'incorrect', 'independ', 'indiana', 'inflat', 'info', 'inquir', 'inquiri', 'insur', 'intermitt', 'inte
rnet', 'inventori', 'investig', 'invoic', 'iowa', 'iraq', 'ish', 'isl', 'israel', 'item', 'jan', 'janu
ari', 'jersey', 'join', 'juli', 'june', 'justifi', 'kansa', 'kentucki', 'kilo', 'kirkland', 'kmart',
'kohl', 'korea', 'kroger', 'kudo', 'labl', 'lark', 'las', 'latin', 'led', 'lesson', 'lightn', 'linda',
'line', 'link', 'lion', 'live', 'llc', 'local', 'locat', 'log', 'loma', 'los', 'loui', 'lowest', 'loya
l', 'loyalti', 'ludicr', 'lug', 'maci', 'mail', 'mainland', 'mall', 'manner', 'march', 'market', 'mark
etplac', 'markup', 'marshal', 'mart', 'martin', 'maryland', 'massachusett', 'math', 'maxx', 'meijer',
'membership', 'merchandis', 'merchant', 'messag', 'metro', 'mexico', 'miami', 'michigan', 'midwest',
'mile', 'militari', 'minnesota', 'mislabel', 'mislead', 'misrepres', 'mississippi', 'missouri', 'monda
y', 'montana', 'mortar', 'multipack', 'nashvill', 'nation', 'nationwid', 'near', 'nearbi', 'nearest',
'neighborhood', 'nespresso', 'netherland', 'netrit', 'nevada', 'nich', 'north', 'northeast', 'northwes
t', 'notif', 'notifi', 'nov', 'novemb', 'nowher', 'nutric', 'nyc', 'obscen', 'obtain', 'oct', 'octob',
'oder', 'offer', 'ohio', 'oklahoma', 'onlin', 'order', 'orderd', 'oregon', 'orlando', 'ouch', 'outda
t', 'outlet', 'outrag', 'overcharg', 'overjoy', 'overpay', 'oversea', 'ozbo', 'pack', 'page', 'paid',
'painless', 'panick', 'passov', 'pay', 'payment', 'pennsylvania', 'perish', 'peru', 'perus', 'petco',
'petfooddirect', 'petsmart', 'petstor', 'pharmaci', 'philadelphia', 'phoenix', 'phone', 'pittsburgh',
'pkgs', 'place', 'pleas', 'polar', 'polici', 'post', 'postag', 'postal', 'price', 'pricei', 'prime',
'prioriti', 'procur', 'prodcut', 'producti', 'profit', 'proflow', 'program', 'prohibit', 'promo', 'pro
mpt', 'psych', 'publix', 'puerto', 'purchac', 'purchas', 'purchs', 'purveyor', 'qti', 'qualifi', 'quan
iti', 'quantiti', 'radius', 'rais', 'ralph', 'ran', 'readili', 'reason', 'rec', 'recd', 'receipt', 're
ceiv', 'reciev', 'reciv', 'reconsid', 'recours', 'rectifi', 'recur', 'redicul', 'reflect', 'refund',
'reimburs', 'reliabl', 'reloc', 'renew', 'rep', 'repackag', 'repli', 'request', 'resel', 'resend', 're
ship', 'resid', 'resolv', 'reson', 'resort', 'resourc', 'respond', 'respons', 'restock', 'retail', 're
think', 'retir', 'rico', 'ridicul', 'ripoff', 'risen', 'riski', 'rite', 'robberi', 'roland', 'rural',
'sacramento', 'sadden', 'safeway', 'sale', 'salli', 'sam', 'saturday', 'save', 'saver', 'saw', 'scam',
'scan', 'scarc', 'schedul', 'scour', 'search', 'section', 'see', 'sell', 'seller', 'sender', 'sept',
'septemb', 'servic', 'shall', 'shelv', 'ship', 'shipe', 'shipment', 'shipper', 'shippment', 'shoddi',
'shop', 'shopp', 'shopper', 'shoprit', 'shore', 'shortag', 'shouldv', 'sign', 'sincer', 'site', 'skyro
cket', 'socal', 'sold', 'someplac', 'sonoma', 'sooner', 'sought', 'southeast', 'specialti', 'specifi',
'speed', 'speedi', 'spoke', 'sporad', 'spotti', 'spree', 'statesid', 'status', 'stock', 'stockpil', 's
toke', 'store', 'street', 'stumbl', 'subcrib', 'submit', 'subscrib', 'subscript', 'suburb', 'supercen
t', 'supermarket', 'supersav', 'superstor', 'supplier', 'suscrib', 'sweden', 'swift', 'tack', 'tact',
'tag', 'target', 'tax', 'teeter', 'temporarili', 'tennesse', 'texa', 'thank', 'thankyou', 'thanx', 'th
ati', 'thrifti', 'thrill', 'thru', 'thursday', 'thx', 'tienda', 'tjs', 'today', 'toronto', 'town', 'tr
ansact', 'trusti', 'trustworthi', 'tucson', 'tuesday', 'turnaround', 'twelv', 'typo', 'unabl', 'unacce
pt', 'unavail', 'unbeat', 'uncertain', 'undamag', 'unfair', 'unfortun', 'unfortunat', 'unit', 'unknow
n', 'unload', 'unreal', 'unreason', 'unreli', 'unsatisfactori', 'unsuccess', 'up', 'updat', 'upscal',
'upstat', 'usa', 'usd', 'usp', 'utah', 'vain', 'vega', 'vender', 'vendor', 'venu', 'verifi', 'via', 'v
ine', 'virginia', 'vitacost', 'von', 'wal', 'walgreen', 'walli', 'walmart', 'warehous', 'warrant', 'wa
shington', 'wayyy', 'web', 'webpag', 'wednesday', 'wegman', 'went', 'west', 'wheatena', 'whim', 'whole
food', 'wholesal', 'whomev', 'william', 'winco', 'wisconsin', 'wish', 'wishlist', 'withdraw', 'worthin
gton', 'written', 'wrote', 'wtf', 'www', 'xmas', 'yay'] :
```

Observations :

1. Here, TruncatedSVD and K-Means is applied on amazon fine food review dataset(~364K).
2. Given dataset is imbalanced in nature (postive reviews:negative reviews = 5.57/1).
3. Only TFIDF featurization is applied on raw text data.
4. Top N features are calulated on the basis of tf-idf score.
5. Co-occurence matrix is calculated with 5 - nearest neighbour.
6. After calculating co-occurrence matrix, we applied TruncatedSVD to retain maximum information, through this step we understood how 'Matrix Factorization' is related to TruncatedSVD technique.
7. Finally, K-Means algorithm is applied to cluster semantically similar words.