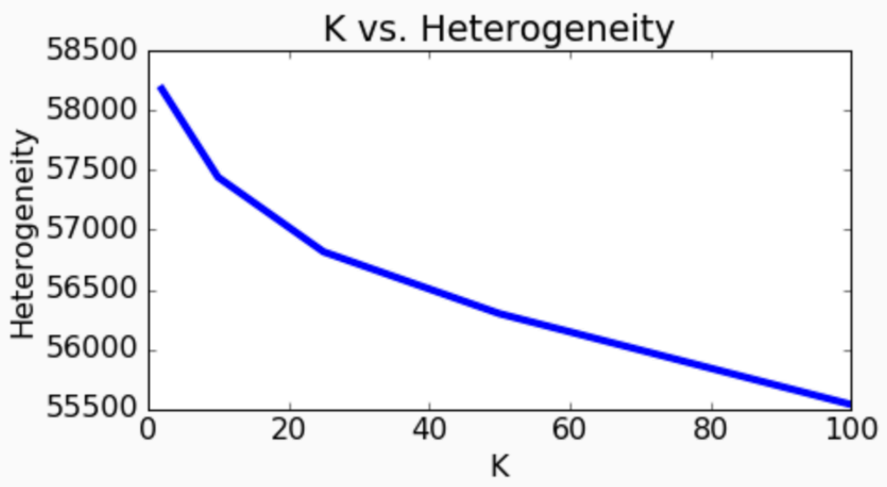


In the above plot we show that heterogeneity goes down as we increase the number of clusters. Does this mean we should always favor a higher K? **Not at all!** As we will see in the following section, setting K too high may end up separating data points that are actually pretty alike. At the extreme, we can set individual data points to be their own clusters (K=N) and achieve zero heterogeneity, but separating each data point into its own cluster is hardly a desirable outcome. In the following section, we will learn how to detect a K set "too large".

Visualize clusters of documents

Let's start visualizing some clustering results to see if we think the clustering makes sense. We can use such visualizations to help us assess whether we have set K too large or too small for a given application. Following the theme of this course, we will judge whether the clustering makes sense in the context of document analysis.

What are we looking for in a good clustering of documents?

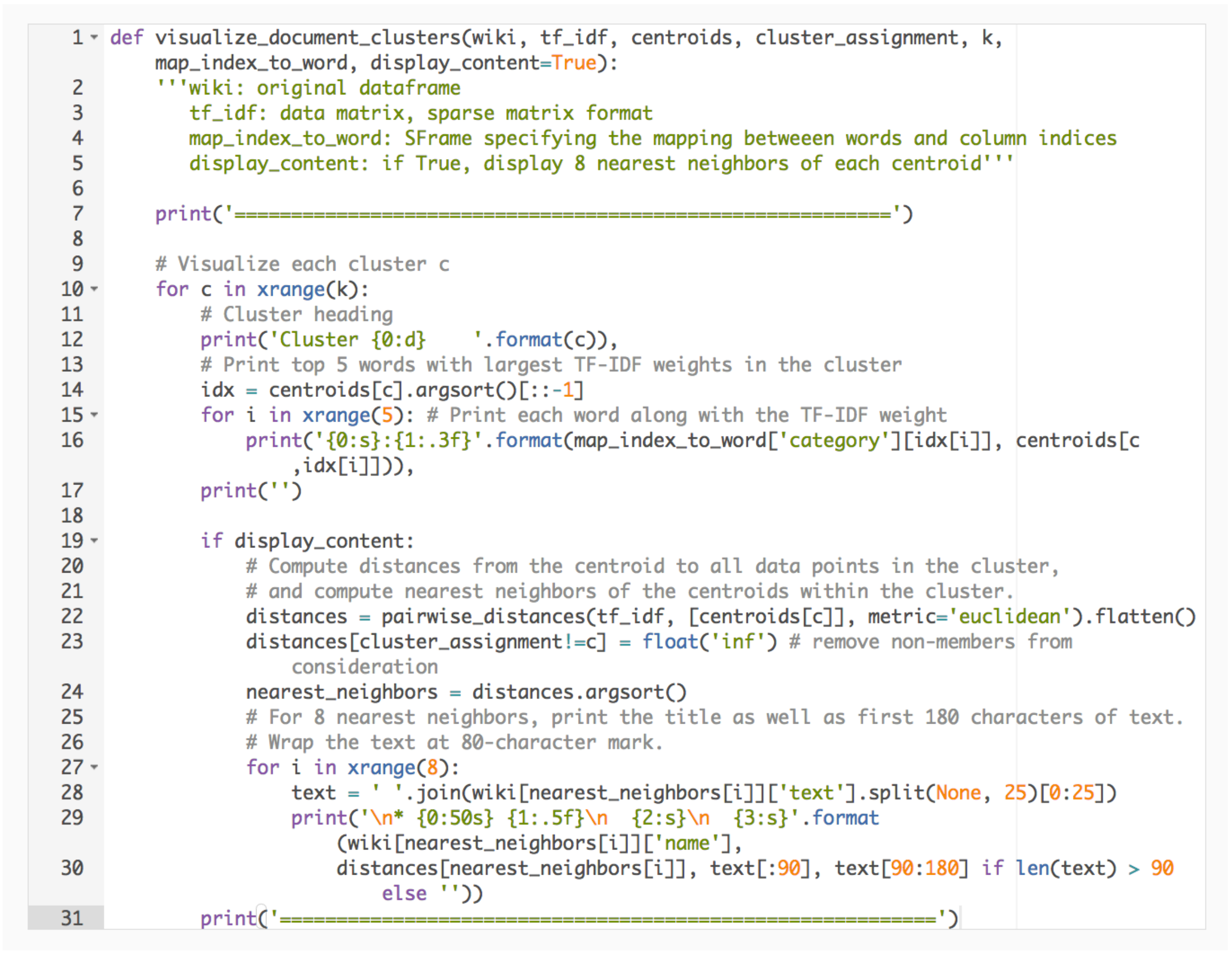
- Documents in the same cluster should be similar.
- Documents from different clusters should be less similar.

So a bad clustering exhibits either of two symptoms:

- Documents in a cluster have mixed content.
- Documents with similar content are divided up and put into different clusters.

To help visualize the clustering, we do the following:

- Fetch nearest neighbors of each centroid from the set of documents assigned to that cluster. We will consider these documents as being representative of the cluster.
- Print titles and first sentences of those nearest neighbors.
- Print top 5 words that have highest tf-idf weights in each centroid.



```python
def visualize_document_clusters(wiki, tf_idf, centroids, cluster_assignment, k, map_index_to_word, display_content=True):
    '''wiki: original dataframe
       tf_idf: data matrix, sparse matrix format
       map_index_to_word: SFrame specifying the mapping betweeen words and column indices
       display_content: if True, display 8 nearest neighbors of each centroid'''

    print('==========================================================')

    # Visualize each cluster c
    for c in xrange(k):
        # Cluster heading
        print('Cluster {0:d}    '.format(c)),
        # Print top 5 words with largest TF-IDF weights in the cluster
        idx = centroids[c].argsort()[::-1]
        for i in xrange(5): # Print each word along with the TF-IDF weight
            print('{0:s}:{1:.3f}'.format(map_index_to_word['category'][idx[i]], centroids[c,idx[i]])),
        print('')

        if display_content:
            # Compute distances from the centroid to all data points in the cluster,
            # and compute nearest neighbors of the centroids within the cluster.
            distances = pairwise_distances(tf_idf, [centroids[c]], metric='euclidean').flatten()
            distances[cluster_assignment!=c] = float('inf') # remove non-members from consideration
            nearest_neighbors = distances.argsort()
            # For 8 nearest neighbors, print the title as well as first 180 characters of text.
            # Wrap the text at 80-character mark.
            for i in xrange(8):
                text = ' '.join(wiki[nearest_neighbors[i]]['text'].split(None, 25)[0:25])
                print('\n* {0:50s} {1:.5f}\n  {2:s}\n  {3:s}'.format(wiki[nearest_neighbors[i]]['name'],
                    distances[nearest_neighbors[i]], text[:90], text[90:180] if len(text) > 90 else ''))
        print('==========================================================')
```

Let us first look at the 2 cluster case (K=2).

```
visualize_document_clusters(wiki, tf_idf, centroids[2], cluster_assignment[2], 2,
    map_index_to_word)
```

Both clusters have mixed content, although cluster 1 is much purer than cluster 0:

- Cluster 0: artists, songwriters, professors, politicians, writers, etc.
- Cluster 1: baseball players, hockey players, football (soccer) players, etc.

Top words of cluster 1 are all related to sports, whereas top words of cluster 0 show no clear pattern.

Roughly speaking, the entire dataset was divided into athletes and non-athletes. It would be better if we sub-divided non-athletes into more categories. So let us use more clusters. How about K=10?

```
k = 10
visualize_document_clusters(wiki, tf_idf, centroids[k], cluster_assignment[k], k,
    map_index_to_word)
```

Cluters 0 and 2 appear to be still mixed, but others are quite consistent in content.

- Cluster 0: artists, poets, writers, environmentalists
- Cluster 1: film directors
- Cluster 2: female figures from various fields
- Cluster 3: politicians
- Cluster 4: track and field athletes
- Cluster 5: composers, songwriters, singers, music producers
- Cluster 6: soccer (football) players
- Cluster 7: baseball players
- Cluster 8: professors, researchers, scholars
- Cluster 9: lawyers, judges, legal scholars

Clusters are now more pure, but some are qualitatively "bigger" than others. For instance, the category of scholars is more general than the category of baseball players. Increasing the number of clusters may split larger clusters. Another way to look at the size of cluster is to count the number of articles in each cluster.

```
np.bincount(cluster_assignment[10])
```

Quiz Question. Which of the 10 clusters above contains the greatest number of articles?

Quiz Question. Which of the 10 clusters contains the least number of articles?

There appears to be at least some connection between the topical consistency of a cluster and the number of its member data points.

Let us visualize the case for K=25. For the sake of brevity, we do not print the content of documents. It turns out that the top words with highest TF-IDF weights in each cluster are representative of the cluster.

```
visualize_document_clusters(wiki, tf_idf, centroids[25], cluster_assignment[25], 25,
                            map_index_to_word, display_content=False) # turn off text for brevity
```

Looking at the representative examples and top words, we classify each cluster as follows. Notice the bolded items, which indicate the appearance of a new theme.

- Cluster 0: composers, songwriters, singers, music producers
- Cluster 1: **poets**
- Cluster 2: **rugby players**
- Cluster 3: baseball players
- Cluster 4: **government officials**
- Cluster 5: football players
- Cluster 6: **radio hosts**
- Cluster 7: **actors, TV directors**
- Cluster 8: professors, researchers, scholars
- Cluster 9: lawyers, judges, legal scholars
- Cluster 10: track and field athletes
- Cluster 11: (mixed; no clear theme)
- Cluster 12: **car racers**
- Cluster 13: **priets, bishops, church leaders**
- Cluster 14: painters, sculptors, artists
- Cluster 15: **novelists**

- Cluster 16: **American football players**
- Cluster 17: **golfers**
- Cluster 18: American politicians
- Cluster 19: **basketball players**
- Cluster 20: **generals of U.S. Air Force**
- Cluster 21: politicians
- Cluster 22: female figures of various fields
- Cluster 23: film directors
- Cluster 24: music directors, composers, conductors

Indeed, increasing K achieved the desired effect of breaking up large clusters. Depending on the application, this may or may not be preferable to the K=10 analysis.

Let's take it to the extreme and set K=100. We have a suspicion that this value is too large. Let us look at the top words from each cluster:

```
k=100
visualize_document_clusters(wiki, tf_idf, centroids[k], cluster_assignment[k], k,
                            map_index_to_word, display_content=False)
# turn off text for brevity -- turn it on if you are curious ;)
```

The class of rugby players have been broken into two clusters (11 and 72). Same goes for soccer (football) players (clusters 6, 21, 40, and 87), although some may like the benefit of having a separate category for Australian Football League. The class of baseball players have been also broken into two clusters (18 and 95).

A high value of K encourages pure clusters, but we cannot keep increasing K. For large enough K, related documents end up going to different clusters.

That said, the result for K=100 is not entirely bad. After all, it gives us separate clusters for such categories as Scotland, Brazil, LGBT, computer science and the Mormon Church. If we set K somewhere between 25 and 100, we should be able to avoid breaking up clusters while discovering new ones.

Also, we should ask ourselves how much **granularity** we want in our clustering. If we wanted a rough sketch of Wikipedia, we don't want too detailed clusters. On the other hand, having many clusters can be valuable when we are zooming into a certain part of Wikipedia.

There is no golden rule for choosing K. It all depends on the particular application and domain we are in.

Another heuristic people use that does not rely on so much visualization, which can be hard in many applications (including here!) is as follows. Track heterogeneity versus K and look for the "elbow" of the curve where the heterogeneity decrease rapidly before this value of K, but then only gradually for larger values of K. This naturally trades off between trying to minimize heterogeneity, but reduce model complexity. In the heterogeneity versus K plot made above, we did not yet really see a flattening out of the heterogeneity, which might indicate that indeed K=100 is "reasonable" and we only see real overfitting for larger values of K (which are even harder to visualize using the methods we attempted above.)

Quiz Question. Another sign of too large K is having lots of small clusters. Look at the distribution of cluster sizes (by number of member data points). How many of the 100 clusters have fewer than 236 articles, i.e. 0.004% of the dataset?

Takeaway

Keep in mind though that tiny clusters aren't necessarily bad. A tiny cluster of documents that really look like each others is definitely preferable to a medium-sized cluster of documents with mixed content. However, having too few articles in a cluster may cause overfitting by reading too much into limited pool of training data.