Script to test feature extraction

Step 0: Setup project directory and dependencies

First we have to construct a citation vector from the co-author column of our dataset

Step 1: Create document term matrix

We use the `tm` package to create a corpus of the title strings.

Step 2.1: Run teacher's clustering algorithm from `kernlab` package

Step 2.2: Run equivalent spectral clustering using a Gaussian-similarity-kernel and k-means clustering.

Here are the details of the implementation:

First, we compute the simmilarity between citations from the TF-IDF or NTF matrix of citations. We use a Gaussian kernel as a measure of similarity. Then, we create an undirected graph based on the similarities to extract some manifold in the data, we thereby obtain A, the affinity matrix. After, we calculate the degree matrix D (diagonal) where each diagonal value is the degree of the respective vertex (*e.g.* sum of rows). We compute the unnormalized graph Laplacian:

$$U = D - A$$

Then, assuming that we want k clusters, we find the k smallest eigenvectors of U. This represents the points in a new k-dimensional space with low-variance. Finally, in this transformed space, it becomes easy for a standard k-means clustering to find the appropriate clusters.

Step 3: Run our spectral clustering algorithm in `lib/SpectralClustering.R`

Here are the details of the implementation:

From the TF-IDF or NTF matrix of citations, we compute the cosine similarity between each citation vectors as follows:

$$similarity = cos(\theta) = \frac{a \cdot b}{||a||\,||b||} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}$$

This matrix is called the **Gram matrix** A. In the first step of the algorithm, we determine the k largest eigenvectors of A: X_k, a n-by-k matrix. Each row of X_k corresponds to a citation vector. Then, we compute the **QR decomposition with column pivoting** applied to ${X_k}^T$, *e.g.* we find the matrices P (permutation matrix, n-by-n), Q (orthogonal, k-by-k) and R (left-upper-triangular, k-by-n), so that:

$${X_k}^T P = QR = Q[R_{11}, R_{12}]$$

R_{11} will be the k-by-k upper-triangular matrix. We then compute the matrix $\hat{R}$:

$$\hat{R} = R_{11}^{-1} R P^T = R_{11}^{-1}[R_{11}, R_{12}]P^T = [I_k, R_{11}^{-1} R_{12}]P^T$$

Finally, the cluster membership of each citation vector is determined by the row index of the largest element in absolute value of the corresponding column of $\hat{R}$.

Step 4: Run evaluation metrics on both methods and compare

```
source(file.path(projDir,"lib",'evaluation_measures.R'))
matching_matrix_sclust <- matching_matrix(authorId,result_sclust)
matching_matrix_sQRclust <- matching_matrix(authorId, result_sQRclust)
matching_matrix_sKMclust <- matching_matrix(authorId, result_sKMclust)
performance_sclust <- performance_statistics(matching_matrix_sclust)
performance_sQRclust <- performance_statistics(matching_matrix_sQRclust)
performance_sKMclust <- performance_statistics(matching_matrix_sKMclust)
```

```
compare_df <- data.frame(method=c("Teacher","QR Spec.C", "Kmeans Spec.C"),
                         precision=c(performance_sclust$precision, performance_sQRclust$precision, perfo
                         recall=c(performance_sclust$recall, performance_sQRclust$recall, performance_sK
                         f1=c(performance_sclust$f1, performance_sQRclust$f1, performance_sKMclust$f1),
                         accuracy=c(performance_sclust$accuracy, performance_sQRclust$accuracy, performa
                         mcc=c(performance_sclust$mcc, performance_sQRclust$mcc, performance_sKMclust$mc
                         time=c(time_sclust, time_sQRclust, time_sKMclust))
kable(compare_df,caption="Comparision of performance for two clustering methods",digits = 2)
```

Table 1: Comparision of performance for two clustering methods

method	precision	recall	f1	accuracy	mcc	time
Teacher	0.26	0.16	0.20	0.72	0.05	1.140998 secs
QR Spec.C	0.18	0.35	0.24	0.52	-0.07	0.174890 secs
Kmeans Spec.C	0.22	0.77	0.34	0.37	0.03	6.499475 secs