Homework #2

Shuang Zhou, N-10746067

November 5, 2015

Part #0

The dataset contains 20 articles in total, collected from http://abcnews.go.com. It involves four categories, namely, Technology, Weather, Entertainment and Sports. From each category there're 5 articles, with their name tagged "X_name.txt", "X" being the initial letter of each category in capital form (e.g. "S_Clippers.txt" is an article originally named "Clippers" in the category "Sports"). This will help identifying the original label of articles to provide possible information for validation.

Part #1

Project Description

The project contains a simplified version of K-Means clustering implemented in Java. The code will be packed with the report, and it's also available at https://github.com/joshua924/PA-HW2. First of all, articles are preprocessed and read in as word vectors. Stop words are removed by looking up a dictionary from Google, then each article is translated into a vector containing count of each word appeared at least once in any of the articles, normalized by the length of each article. To find similar articles, three distance measurement algorithms are used for performance comparison. And as part of K-Means algorithm, the termination criteria is that all clusters remain the same after an iteration. For initialization, we set the number of starting instances to 4, because this is intuitively the best choice as we picked four categories of articles.

Just for the record, I didn't use any external code sources, all code is written by me for this project.

Distance Measurement Algorithms

Euclidean distance, Min-hash distance and Cosine distance are used in this project. Below is a brief introduction of how they work for this project.

Euclidean distance

Euclidean distance is straightforward, it takes two vectors, $A = [a_1, a_2, ..., a_n]$ and $B = [b_1, b_2, ..., b_n]$ then compute

$$\sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

Min-hash distance

Let h be a hash function that maps the members of A and B to distinct integers, and for any set S define $h_{min}(S)$ to be the minimal member of S with respect to h that is, the member x of S with the minimum value of $h(x)$. Now, if we apply h_{min} to both A and B, we will get the same value exactly when the element of the union $A \cup B$ with minimum hash value lies in the intersection $A \cap B$. The probability of this being true is the ratio above, and therefore:

$$Pr[h_{min}(A) = h_{min}(B)] = J(A, B)$$

That is, the probability that $h_{min}(A)$ = $h_{min}(B)$ is true is equal to the similarity $J(A, B)$, assuming randomly chosen sets A and B. In other words, if r is the random variable that is one when $h_{min}(A)$ = $h_{min}(B)$ and zero otherwise, then r is an unbiased estimator of $J(A, B)$. r has too high a variance to be a useful estimator for the Jaccard similarity on its own it is always zero or one. The idea of the MinHash scheme is to reduce this variance by averaging together several variables constructed in the same way.

Considering the values in word vectors are of type double, we adopt a slightly modified version of Jaccard Similarity to compute. For each "intersection" we have between two vectors, we compute $a_i * b_i$ as the increment to our numerator.

Cosine distance

This is a simple variation of cosine similarity, given two vectors $A = [a_1, a_2, ..., a_n]$ and $B = [b_1, b_2, ..., b_n]$, the cosine similarity

$$S(A, B) = \frac{A \times B}{|A| \times |B|}$$

in which $A \times B$ is the inner product of two vectors. Then the cosine distance $D(A, B)$ can be computed by

$$D(A, B) = 1 - S(A, B)$$

Results and Performances

We run the program with different algorithms, to evaluate the performance of each algorithm, I define the error rate in this way:

$$E(a) = \sum_{j=1}^{20} isFalse(C_j = L_j)$$

where a is the algorithm we use and C_j is the cluster it assigns to instance j and L_j is the original label of j, and the label of C_j is manually decided. We conclude all performance in Table 1.

Table 1. Performance Comparison

	Homemade	RapidMiner
Euclidean Distance	0.55	0.45
Cosine Distance	0.30	0.15
Min Hash Distance	0.35	0.5

Comparing the homemade version of algorithms against RapidMiner, most of them is not as accurate, this can be caused by multiple reasons: optimized selection of starting instances, number of iterations, and maybe some other tricks.

Comparing different algorithms, we can see that K-Means with Cosine Similarity performs best out of these three distance measurements. This is actually not surprising, as cosine similarity is one of the most popular distance algorithms to measure similarities of documents in Information Retrieval (in Space Vector Model).

PART #2