Concept decompositions for large sparse test data using clustering

Idea of this project is to cluster collection of text data using k-means algorithm on vector space model. Specifically, we interpret n text documents as d-dimensional vectors of certain weighted word frequencies which we cluster. Using obtained results we approximate corresponding word-by-document matrix.

We distribute our project goals over 3 main parts: Vector space model creation, Cluster construction, Decomposition analysis.

Vector space model creation

There are 2 main subtasks in order to create a vector space model:

[1] Data extraction

[2] Model creation

[1] We used unordered\_map container from C++ std package to build the hash-table of depth 2 for storing words and vectors of their occurrence (matrix row). This container shows the best performance over the number of words and search queries we use to count them all.

There are 5 main steps in our data extraction, given bellow in respective order:

1. Extracting all unique words from a set of documents

(ignoring case, ignoring non-alphabetic terms)

2. Eliminate non-content-bearing ‘stop-words’

(list of stop-words we used can be found in Frakes and Baeza-Yates, 1992, Chapter 7)

3. Stemming words to their root

(we used a Porter stemmer found in Frakes and Baeza-Yates, 1992, Chapter 8)

4. For each document, count the number of occurrences of given word

(full benefits of unordered\_map were exploited)

5. Eliminate high/low-frequency words

(this can be chosen in user interface)

[2] After Data Extraction, only d words remain and we continue by creating a d\*n word-by-document matrix. After performing txn (normalized term frequency) algorithm on the previously mentioned matrix, we have normalized document vectors which we use in k-means algorithm that follows.

Cluster construction

[1] Differences between spherical and classic k-means clustering:

1. All vectors have all positive coordinates.

2. All vectors have the norm 1. (They lie on a sphere with center at coordinate system origin

with radius 1.)

[2] We had to implement objective function that measures the quality of current clustering.

[3] We have to find initial partition or initial concept vector. We used two different techniques to find it.

1. Choose random partition to equally sized parts.

2. Calculate the total concept vector and obtain k starting concept vectors by randomly

perturbing this vector.

[4] Then we perform spherical k-means algorithm which consists of these steps (2., 3. and 4. in a loop):

1. Start with some concept vectors that you previously got.

2. For each document vector find the closest concept vector and use the closest concept

vectors to create new partition of document vectors.

3. Compute the new concept vector of each set in a partition. Its norm must be 1.

4. Check if some “stopping criterion” is met. We used limited number of loop iterations and

checking if objective function increased. If none of them is met go to step 2.

Decomposition analysis

Using the centroids and clusters, we create a new matrix which approximates matrix from the beginning. This process is called - concept decomposition – computing the least-squares approximation onto linear subspace spanned by the concept vectors.

We calculated another approximation by k-truncated SVDs to compare which decomposition approximates the matrix better. We measure the error in approximating using the Frobenius norm of the difference matrix.

RESULTS:

We test our work on a set of text-books and fantastic novels to show how they are correctly clustered. After the input

[partitions\_vector, error\_cd, error\_tSVD]=matLabMain(3,100),

we get the output

partitions\_vector=[3 1 3 1 3 3 2 2 2 2 1 2 3 3] ; error\_cd=3.0512 ; error\_tSVD=3.6529

where each of the number in partitions\_vector represents a cluster to which the i-th (i=1,...d) document belongs to. From the error\_cd – error of concept decomposition approximation and error\_tSVD – error of truncated SVDs approximation, we can see that the concept decomposition gave the better approximation.