1 Clustering of wikipedia articles

1.1 Obtaining a sparse matrix representation

The following tasks were carried out to tokenize the wikipedia dump into a sparse matrix consisting of (word, count) pairs for suitably selected features, the tasks were carried out using python and the hadoop streaming api.

(a) Word count: Mappers for the word count task output (word, count) pairs for the portion of the file allocated to them, the reducer (single node) aggregates the counts. The reducer sorts words in descending order by frequency, the top 50000 words in the dictionary generated were chosen to be features. Power law effects are useful for feature selection, the distribution of word counts follows a power-law and has a heavy tail, we eliminated words occurring less than 5 times while computing the feature dictionary. Tokenizing was done by stripping punctuation characters representing mark-up.

Remark: We also successfully completed the first task i.e. word counts in Java by using the Lucene Wikipedia Tokenizer and mahout’s XmlRecordReader (but eventually decided to stick with python for the rest of homework).

(b) Sparse matrix generation: The task can be carried out by multiple mappers, we used a script for recovering title and category information and filtering words found in the article text as opposed to the references. The script ran on hadoop file system on a local machine with 5 mappers, although it failed on the cluster with 551 mappers (presumably because of issues with xml parsing). We instead ran the script on offloaded dump on the local machine, achieving the same results.

1.2 Clustering

We implemented a spectral clustering algorithm that first projects the sparse feature vectors (weighted by the $tf - idf$ scores) onto the space spanned by the top $t$ singular vectors of the sparse document-term matrix followed by the $k$-means algorithm on the reduced feature space.

The following steps were involved in the design, the clustering was performed on a local machine:

(a) Compute the number of documents in the corpus that each feature occurs in, the weight of word $w$ in document $d$ is adjusted to $\frac{tf(w,d)}{idf(w)}$ on the fly. Words not in the webster dictionary are eliminated to reduce the feature set size to 22000.

(b) The number of documents 1.7 million (filtered to remove redirects and others with small number of features) is too large for direct $SVD$ computation. Further, having a feature size of 20000 greatly reduces the speed of the $k$-means algorithm to be used subsequently. The paper [1] addresses this problem and provides a feature reduction method that drastically reduces the matrix dimensions while preserving accuracy.

Roughly matrix rows and columns should be sampled with probability proportional to the norm, that is documents with a large number of features get chosen with high probability. We restricted the document set to documents for which the number of features is above a certain threshold, and sampled features that had high norm with respect to the chosen document set and were frequent in the corpus.
(c) The SVD was computed for the reduced matrix from step (2). The dimension of the projected space \( t \) was chosen to be such that 
\[
\sum_{i \in [k]} \sigma_i^2 = 0.8,
\]
that is 0.9 fraction of the total variance is captured by projection onto the span of the first \( t \) singular vectors.

The magnitudes of the singular values decay smoothly and have a heavy tail, so a value of \( t \approx 200 \) suffices, the space spanned by the top \( t \) singular vectors has low dimension and captures most of the variance making spectral clustering much more efficient than random projection.

(d) Feature vectors were projected onto the \( t \) dimensional subspace from step (3), seeds were chosen by randomly sampling articles from the corpus. The number of clusters was chosen by binary search, optimizing the average mean squared error over the corpus. The performance improved with increasing the number of clusters and then stabilized at \( k = 200 \).

Hierarchical clustering can be performed by repeating the steps (a)-(d) restricted to the documents found in a \( k \) means cluster.

1.3 Results

Clustering a subset of the corpus chosen where the articles have a large number of feature vectors yields clusters that match real world topics very well. There are 200 clusters at the first level, here are some clusters sampled at random along with articles present in them. The format is cluster number, topic inferred manually, list of articles:

(a) Cluster 198, Soccer: arsenal f c, afc ajax, a s roma, australian rules football, acf fiorentina, bobby charlton, brisbane broncos, diego maradona, david beckham, essendon football club...

(b) Cluster 74, Movies: academy award, anthony hopkins, adrian lamo, charlie chaplin, cecil b demille, cinema of italy, fritz lang, federico fellini, gene kelly...

(c) Cluster 168, Nuclear bombs: cold fusion,einsteinium, enrico fermi, freeman dyson,fallout shelter, fat man,greenpeace, glenn t seaborg, intercontinental ballistic missile,international atomic energy agency,james lovelock,little boy...

(d) Cluster 43, Wildlife: algae, asparagales, allosaurus, albertosaurus, acacia, amaranth, ant, bird, bryozoa, brain, bluetongue disease, bipedalism, bee, zebrafish, carnivora...

(e) Cluster 40, special days: columbus day, flag of estonia, bonfire, millerism, maya calendar, japanese festivals, japanese new year, value at risk, benny hinn, saint joseph s day, christmas eve, earth day, shrove tuesday, cecil day lewis, shopping hours...

Most of the clusters obtained are coherent and correspond to a wikipedia category or can be classified easily manually. A file containing clusters for a subset of the corpus is attached.

References