11-741 k-means Clustering

Xin Qian

September 19, 2016

Due: Sept 22, 2016 11:59 PM

1 Corpus Exploration

Dataset	Development set	Test set
Total number of documents	942	942
Total number of words	254852	249516
Total number of unique words	14063	13924
Average number of unique words per document	174.59	173.37

For the first document in the development set, the total number of unique words is 161. All twice occurrence word ids are, 2, 5, 10, 18, 23, 27, 28, 30, 32, 42, 44, 45, 46, 50, 52, 60, 62, 69, 79, 87, 91, 99, 102, 107, 114, 141.

2 Experiments

The number of document clusters and stopping criteria are the same for all three approaches. The number of document clusters is 67. Stopping criteria is either cluster labels does not change for all document instances after an assignment-update iteration, or the difference between two iterations on sum of cosine similarity for all document instances differs no more than 1e-4. Usually the algorithm stops within 10 iterations.

2.1 Baseline approach

Parameters are tuned at two stages, first coarsely and then fine-grained.

k=67. The mean is 0.541058776344, variance is 0.0007055614969. The best F1 out of 10 is 0.579381737382.

2.2 k-means++ approach

k=67. The mean is 0.551529967305, variance is 0.000512666340166. The best F1 out of 10 is 0.598781761183.

2.3 Custom algorithm

Keeping the kmeans++ initialization and kmeans iteration with cosine similarity scheme unchanged, my custom algorithm mainly replaces each term frequency in the term-document matrix with the

tf-idf value.

$$tf - idf_{t,d} = tf_{t,d} \cdot \log \frac{N}{df_t}$$

This custom algorithm solves the problem in raw term frequency scheme where all terms are treated equal, however, it is not a proper way since some terms are less informative e.g. (stop words) or discriminative in contributing to the relevance score. For example, a collection of newswire is likely to have the term *report* appears many times. We might want to attenuate the unwanted contribution that frequent terms make in scoring. Since we are discriminate documents in ranking, document frequency would be a better choice than the collection frequency (which counts multiple occurrences of a term.) Therefore it is a good choice to multiply the raw term frequency with the inverse document frequency, where taking log dampens the scaling down effect. I'm therefore trying to improve upon external metrics, e.g. F-measure, accuracy, NMI and purity, to improve the clustering quality and convergence rate to the underlining real class distribution.

k=67. The mean is 0.606431487906, variance is 0.000777387170029. The best F1 out of 10 is 0.658362769998.

2.4 Analysis

The custom algorithm (with tf-idf weighting, kmeans++ centroid initialization and cosine similarity measurement) achieves the best results. The baseline k-means performed worst. For the custom algorithm, the tf-idf weighting scheme worked out well and didn't increase much on computation cost. This improvement is desirable in that it emphasizes the informativeness of rare term while suppress the expressiveness of frequent terms. This follows the intuition that a term that occurs many times only within a small collection of documents must have high discriminating power to these documents v.s. others, which therefore better models the 1-versus-all classification problem or the clustering problem.

2.5 Software implementation & data preprocessing

2.6 Source code and test set results

Please find the README.txt file for instructions on running the code.

Please also find the xinq-test-clusters.txt file as the final document clustering result on the test set. Thanks for your support!