Computing Social Score of Web Artifacts

Soumyajit Ganguly Simran Kedia
Jaspreet Kaur J N Venkatesh

Lakshminarayanan Srinivasan
Prof. Vasudeva Varma
IIIT Hyderabad

April 17, 2015

Abstract

Different social sites have different entities, like Facebook has posts, Twitter has tweets, Youtube has videos, Pinterest has pints, etc, etc. Each of these sites have different metrics to calculate a social (popularity) score. The task is to compute a single metric which takes into account the different metrics from different social media sites.

1 Introduction

Web artifacts are items like news articles, web pages, videos, pints, tweets, facebook posts or any other URLs. Each of these sites have different metrics to calculate a popularity score, eg: for Facebook, it would be number of likes, shares and comments; for Twitter it would be number of favourites, retweets, comments, etc, etc.

The challenge is to find the social/popularity score of an item, if it has been shared on multiple places. For example, the task is to find the popularity score of a video URL shared on facebook, twitter, youtube, etc.

So, we have to aggregate similar entities from multiple sources and compute a combined score obtained for each of the artifacts from extracted data.

2 Approach

To start with, we decided to work only on Facebook and Twitter data sets. We first started building a naive model, where the social score of an artifact would just be the sum total of number of likes and shares of Facebook posts and number of shares and favorites of Twitter tweets containing that artifact.

But, as we can see this is not a correct measure of the popularity of an artifact. There may be tweets/posts talking about an artifact and may not always quote the artifact and talk about it 1. kNN - first creating histograms for every document using count vectorizer - normalize the histograms by constructing tf-idf - we now have vector representations for each document - for all the artifacts (train data) we take all the vectors and train the k-NN classifier in an unsupervised model. Default parameters. - Take the non-artifact posts, convert them to respective feature representations using the same vocabulary as in training. - Get the top-10 closest neighbors for each non-artifact and the respective similarity scores.

2. Lucene - Take all the artifacts and put them in directory for indexing. - Indexing is done for the various files with artifacts and the indexed structures are stored in a separate directory. - All non artifacts are then queried against the indexed files and 10 most closest documents per query are retrieved. - Then respective score of the non-web artifacts are calculated.

3 Experiments

Lucene: -All the text from non web artifacts is dumped to a file,which is then made noise free and further split into several files to get a no of files corresponding to various artifacts. -These files are then passed as input to lucene to let them get indexed. -All the indexed data is stored in a separate folder. -Then for those artifacts in which there were no links,the artifacts data get queried so as to find similarity of them with other documents. -Top 10 similar documents are retrieved.

4 Results

5 References

1. Christopher D. Manning, Prabhakar Raghavan, Hinrich Schutze. Introduction to Information Retrieval, 2008

2. Charu C. Aggarwal, ChengXiang Zhai. Mining Text Data, *Springer*, 2012

3. Pedregosa et al. Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research 12, pp. 2825-2830*, 2011

4. Bentley, J.L. Multidimensional binary search trees used for associative searching, *Communications of the ACM*, 1975

5. Omohundro S.M, Five ball-tree construction algorithms, *International Computer Science Institute Technical Report*, 1989

6. Bawa, M., Condie, T., Ganesan, P., LSH Forest: Self-Tuning Indexes for Similarity Search, *Proceedings of the 14th international conference on World Wide Web*, 2005

7. Jiangong Zhang, Xiaohui Long, Torsten Suel, Performance of Compressed Inverted List Caching in Search Engines, *17th International Conference on World Wide Web*, 2008

8. Gianni Amati and Cornelis Joost Van Rijsbergen, Probabilistic models of information retrieval based on measuring the divergence from randomness, *ACM Transactions on Information Systems*, 2002

9. Yusuke Yamamoto, Java library for the Twitter API, *http://www.twitter4j.org/*, 2007

10. Massimo Di Pierro, web2py for Scientific Applications, *Computing in Science and Engineering, vol.13, no. 2, pp. 64-69*, 2011