T^2K^2: The Twitter Top-K Keywords Benchmark

Ciprian-Octavian Truică[1,a], Jérôme Darmont[2,b]

[1]Computer Science and Engineering Department, Faculty of Automatic Control and Computers, University Politehnica of Bucharest, Bucharest, Romania
[2]Université de Lyon, Lyon 2, ERIC EA 3083, France
[a]ciprian.truica@cs.pub.ro, [b]jerome.darmont@univ-lyon2.fr

Abstract. Information retrieval from textual data focuses on the construction of vocabularies that contain weighted term tuples. Such vocabularies can then be exploited by various text analysis algorithms to extract new knowledge, e.g., top-k keywords, top-k documents, etc. Top-k keywords are casually used for various purposes, are often computed on-the-fly, and thus must be efficiently computed. To compare competing weighting schemes and database implementations, benchmarking is customary. To the best of our knowledge, no benchmark currently addresses these problems. Hence, in this paper, we present a top-k keywords benchmark, T^2K^2, which features a real tweet dataset and queries with various complexities and selectivities. T^2K^2 helps evaluate weighting schemes and database implementations in terms of computing performance. To illustrate T^2K^2's relevance and genericity, we show how to implement the TF-IDF and Okapi BM25 weighting schemes, on one hand, and relational and document-oriented database instantiations, on the other hand.

Keywords: Top-k keywords, Benchmark, Term weighting, Database systems

1 Introduction

Analyzing textual data is a current challenge, notably due to the vast amount of text generated daily by social media. One approach for extracting knowledge is to infer from texts the top-k keywords to determine trends [1,14], or to detect anomalies or more generally events [7]. Computing top-k keywords requires building a weighted vocabulary, which can also be used for many other purposes such as topic modeling and clustering. Term weights can be computed at the application level, which is inefficient when working with large data volumes because all information must be queried and processed at a layer different from storage. A presumably better approach is to process information at the storage layer using aggregation functions, and then return the result to the application layer. Yet, the term weighting process remains very costly, because each time a query is issued, at least one pass through all documents is needed.

To compare combinations of weighting schemes, computing strategies and physical implementations, benchmarking is customary. However, to the best of our knowledge, there exists no benchmark for this purpose. Hence, we propose

in this paper the Twitter Top-K Keywords Benchmark (T^2K^2), which features a real tweet dataset and queries with various complexities and selectivities. We designed T^2K^2 to be somewhat generic, i.e., it can compare various weighting schemes, database logical and physical implementations and even text analytics platforms [18] in terms of computing efficiency. As a proof of concept of T^2K^2's relevance and genericity, we show how to implement the TF-IDF and Okapi BM25 weighting schemes, on one hand, and relational and document-oriented database instantiations, on the other hand.

The remainder of this paper is organized as follows. Section 2 reviews text-oriented benchmarks. Section 3 provides T^2K^2's generic specification. Section 4 details T^2K^2's proof of concept, i.e., its instantiation for several weighting schemes and database implementations. Finally, Section 5 concludes this paper and hints at future research.

2 Related Works

Term weighting schemes are extensively benchmarked in sentiment analysis [15], semantic similarity [11], text classification and categorization [8,9,11,13], and textual corpus generation [19]. Benchmarks for text analysis focus mainly on algorithm accuracy, while either term weights are known before the algorithm is applied, or their computation is incorporated with preprocessing. Thus, such benchmarks do not evaluate weighting scheme construction efficiency as we do.

Other benchmarks evaluate parallel text processing in big data applications in the cloud [4,5]. PRIMEBALL notably specifies several relevant properties characterizing cloud platforms [4], such as scale-up, elastic speedup, horizontal scalability, latency, durability, consistency and version handling, availability, concurrency and other data and information retrieval properties. However, PRIMEBALL is only a specification; it is not implemented.

3 T^2K^2 Specification

Typically, a benchmark is constituted of a data model (conceptual schema and extension), a workload model (set of operations) to apply on the dataset, an execution protocol and performance metrics [3]. In this section, we provide a conceptual description of T^2K^2, so that it is generic and can cope with various weighting schemes and database logical and physical implementations.

3.1 Data Model

The base dataset we use is a corpus of 2 500 000 tweets that was collected using Twitter's REST API to read and gather data. Moreover, we applied preprocessing steps to the raw corpus to extract the additional information needed to build

a weighted vocabulary: 1) extract all tags and remove links; 2) expand contractions, i.e., shortened versions of the written and spoken forms of a word, syllable, or word group, created by omission of internal letters and sounds [2], e.g., "it's" becomes "it is"; 3) extract sentences and remove punctuation in each sentence, creating a clean text; 4) for each sentence, extract lemmas and create a lemma text; 5) for each lemma t in tweet d, compute the number of co-occurrences $f_{t,d}$ and term frequency $TF(t,d)$, which normalizes $f_{t,d}$.

T^2K^2 database's conceptual model (Figure 1) represents all the information extracted after the text preprocessing steps. Information about tweet *Author* are a unique identifier, first name, last name and age. Information about author *Gender* is stored in a different entity to minimize the number of duplicates of gender type. *Documents* are identified by the tweet's unique identifier and store the raw tweet text, clean text, lemma text, and the tweet's creation date. *Writes* is the relationship that associates a tweet to its author. Tweet location is stored in the *Geo_Location* entity to avoid duplicates again. *Word* bears a unique identifier and the actual lemma. Finally, weights $f_{t,d}$ and $TF(t,d)$ for each lemma and each document are stored in the *Vocabulary* relationship.

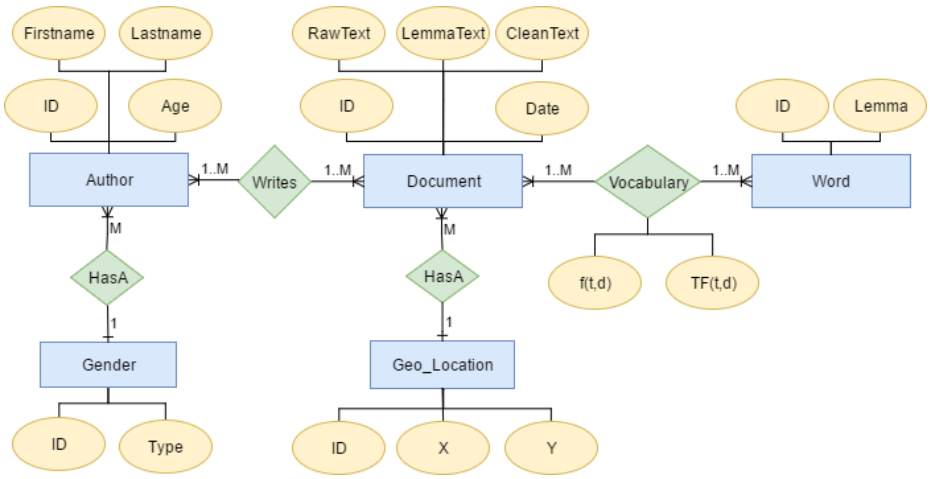

Fig. 1: T^2K^2 Conceptual Data Model

The initial 2 500 000 tweet corpus is split into 5 different datasets that all keep an equal balance between the number of tweets for both genders, location and date. These datasets contain 500 000, 1 000 000, 1 500 000, 2 000 000 and 2 500 000 tweets, respectively. They allow scaling experiments and are associated to a scale factor (SF) parameter, where $SF \in \{0.5, 1, 1.5, 2, 2.5\}$, for conciseness sake.

3.2 Workload Model

The queries used in T^2K^2 are designed to achieve two goals: 1) compute different term weighting schemes using aggregation functions and return the top-k keywords; 2) test the performance of different database management systems. T^2K^2 queries are sufficient for achieving these goals, because they test the query execution plan, internal caching and the way they deal with aggregation. More precisely, they take different group by attributes into account and aggregate the information to compute weighting schemes for top-k keywords.

T^2K^2 features four queries $Q1$ to $Q4$ that compute top-k keywords w.r.t. constraint(s): $c_1(Q1)$, $c_1 \wedge c_2(Q2)$, $c_1 \wedge c_3(Q3)$, $c_1 \wedge c_2 \wedge c_3(Q4)$. c_1 is *Gender.Type* = *pGender*, where parameter *pGender* $\in$ {*male, female*}. c_2 is *Document.Date* $\in$ *[pStartDate, pEndDate]*, where *pStartDate, pEndDate* $\in$ *[2015-09-17 20:41:35, 2015-09-19 04:05:45]* and *pStartDate* $<$ *pEndDate*. c_3 is *Geo_location.X* $\in$ *[pStartX, pEndX]* and *Geo_location.Y* $\in$ *[pStartY, pEndY]*, where *pStartX, pEndX* $\in$ *[15, 50]*, *pStartX* $<$ *pEndX*, *pStartY, pEndY* $\in$ *[-124, 120]* and *pStartY* $<$ *pEndY*. Queries bear different levels of complexity and selectivity.

3.3 Performance Metrics and Execution Protocol

We use each query's response time $t(Q_i)$ as metrics in T^2K^2. Given scale factor SF, all queries $Q1$ to $Q4$ are executed 40 times, which is sufficient according to the central limit theorem. Average response times and standard deviations are computed for $t(Q_i)$. All executions are warm runs, i.e., either caching mechanisms must be deactivated, or a cold run of $Q1$ to $Q4$ must be executed once (but not taken into account in the benchmark's results) to fill in the cache. Queries must be written in the native scripting language of the target database system and executed directly inside said system using the command line interpreter.

4 T^2K^2 Proof of Concept

In this section, we aim at illustrating how T^2K^2 works and at demonstrating that it can adequately benchmark what it is designed for, i.e., weighting schemes and database implementations. For this sake, we first compare the TF-IDF and Okapi BM25 weighting schemes in terms of computing efficiency. Second, we seek to determine whether a document-oriented database is a better solution than in a relational databases when computing a given term weighting scheme.

4.1 Weighting Schemes

Let D be the corpus of tweets, $N = |D|$ the total number of documents (tweets) in D and n the number of documents where some term t appears. The TF-IDF weight is computed by multiplying the augmented term frequency $TF(t,d) = K + (1-K) \cdot \frac{f_{t,d}}{\max_{t' \in d}(f_{t',d})}$) by the inverted document frequency $IDF(t,D) =$

$1 + \log \frac{N}{n}$, i.e., $TFIDF(t, d, D) = TF(t, d) \cdot IDF(t, D)$. The augmented form of TF prevents a bias towards long tweets when the free parameter K is set to 0.5 [12]. It uses the number of co-occurrences $f_{t,d}$ of a word in a document, normalized with the frequency of the most frequent term t', i.e., $\max_{t' \in d}(f_{t',d})$.

The Okapi BM25 weight is given in Equation (1), where $||d||$ is d's length, i.e., the number of terms appearing in d. Average document length $avg_{d' \in D}(||d'||)$ is used to remove any bias towards long documents. The values of free parameters k_1 and b are usually chosen, in absence of advanced optimization, as $k_1 \in [1.2, 2.0]$ and $b = 0.75$ [10,16,17].

$$Okapi(t, d, D) = \frac{TFIDF(t, d, D) \cdot (k_1 + 1)}{TF(t, d) + k_1 \cdot (1 - b + b \cdot \frac{||d||}{avg_{d' \in D}(||d'||)})} \tag{1}$$

The sum $S\_TFIDF(t, d, D) = \sum_{i=1}^{N} TFIDF(t, d_i, D)$ of all TF-IDFs and the sum $S\_Okapi(t, d, D) = \sum_{i=1}^{N} Okapi(t, d_i, D)$ of all Okapi BM25 weights constitute the term's weights that are used to construct the list of top-k keywords.

4.2 Relational Implementations

Database The logical relational schema used in both relational databases management systems (Figure 2) directly translates the conceptual schema from Figure 1.

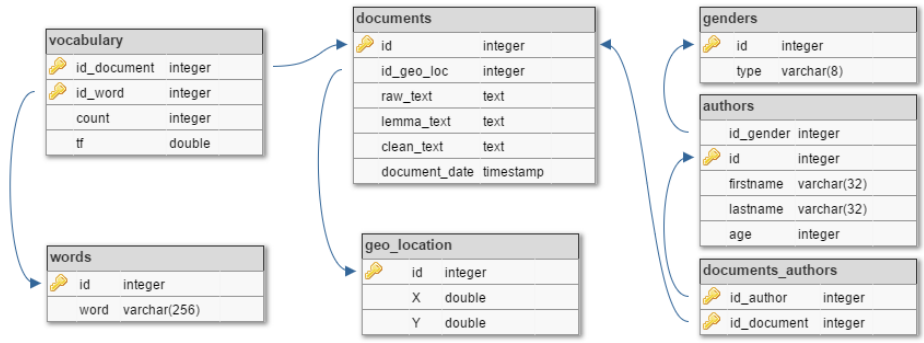

Fig. 2: T^2K^2 Relational Logical Schema

Queries Text analysis deals with discovering hidden patterns from texts. In most cases, it is useful to determine such patterns for given groups, e.g., males and females, because they have different interests and talk about disjunct subjects. Moreover, if new events appear, depending on the location and time of day,

these subject can change for the same group of people. The queries we propose aim to determine such hidden patterns and improve text analysis and anomaly detection.

Let us express T^2K^2's queries in relational algebra. c_1, c_2 and c_3 are the constraints defined in Section 3.2, adapted to the relational schema.

$Q1 = \gamma_L(\pi_{documents.id,words.word,f_w(vocabulary.count,vocabulary.tf)}(\sigma_{c_1}(documents \bowtie_{c_4} documents\_authors \bowtie_{c_5} authors \bowtie_{c_6} genders \bowtie_{c_7} vocabulary \bowtie_{c_8} words)))$, where c_4 to c_8 are join conditions; f_w is the weighting function that computes TF-IDF or Okapi BM25, which takes two parameters: $vocabulary.count = f_{t,d}$ and $vocabulary.tf = TF(t,d)$; γ_L is the aggregation operator, where $L = (F, G)$, with $F = \text{sum}(f_w(vocabulary.count, vocabulary.tf))$ and G is the $words.word$ attribute that appears in the group by clause.

$Q2 = \gamma_L(\pi_{documents.id,words.word,f_w(vocabulary.count,vocabulary.tf)}(\sigma_{c_1 \wedge c_2}(documents \bowtie_{c_4} documents\_authors \bowtie_{c_5} authors \bowtie_{c_6} genders \bowtie_{c_7} vocabulary \bowtie_{c_8} words)))$.

$Q3 = \gamma_L(\pi_{documents.id,words.word,f_w(vocabulary.count,vocabulary.tf)}(\sigma_{c_1 \wedge c_3}(documents \bowtie_{c_4} documents\_authors \bowtie_{c_5} authors \bowtie_{c_6} genders \bowtie_{c_7} vocabulary \bowtie_{c_8} words \bowtie_{c_9} geo\_location)))$, where c_9 is the join condition between *documents* and *geo_location*.

$Q4 = \gamma_L(\pi_{documents.id,words.word,f_w(vocabulary.count,vocabulary.tf)}(\sigma_{c_1 \wedge c_2 \wedge c_3}(documents \bowtie_{c_4} documents\_authors \bowtie_{c_5} authors \bowtie_{c_6} genders \bowtie_{c_7} vocabulary \bowtie_{c_8} words \bowtie_{c_9} geo\_location)))$.

4.3 Document-oriented Implementation

Database In a Document Oriented Database Management System (DODBMS), all information is typically stored in a single collection. The many-to-many *Vocabulary* relationship from Figure 1 is modeled as a nested document for each record. The information about user and date become single fields in a document, while the location becomes an array. Figure 3 presents an example of the DODBMS document.

Queries In DODBMSs, user-defined (e.g., JavaScript) functions are used to compute top-k keywords. The TF-IDF weight can take advantage of both native database aggregation (NA) and MapReduce (MR). However, due to the multitude of parameters involved and the calculations needed for the Okapi BM25 weighting scheme, the NA method is usually difficult to develop. Thus, we recommend to only use MR in benchmark runs.

5 Conclusion

Jim Gray defined four primary criteria to specify a "good" benchmark [6]. *Relevance:* The benchmark must deal with aspects of performance that appeal to the largest number of users. Considering the wide usage of top-k queries in

```
{    _id : 644626677310603264,
     rawText : "Amanda's car is too much for my headache",
     cleanText : "Amanda is car is too much for my headache",
     lemmaText : "amanda car headache",
     author : 970993142,
     geoLocation : [ 32, 79 ],
     gender : "male",
     age : 23,
     lemmaTextLength : 3,
     words : [ { "tf" : 1, "count" : 1, "word" : "amanda"},
               { "tf" : 1, "count" : 1, "word" : "car" },
               { "tf" : 1, "count" : 1, "word" : "headache"} ],
     date : ISODate("2015-09-17T23:39:11Z") }
```

Fig. 3: Sample DODBMS Document

various text analytics tasks, we think T^2K^2 fulfills this criterion. We also show in Section 4 that our benchmark achieves what it is designed for.

Portability: The benchmark must be reusable to test the performances of different database systems. We successfully instantiated T^2K^2 within two types of database systems, namely relational and document-oriented systems.

Simplicity: The benchmark must be feasible and must not require too many resources. We designed T^2K^2 with this criterion in mind (Section 3), which is particularly important for reproducibility. We notably made up parameters that are easy to setup.

Scalability: The benchmark must adapt to small or large computer architectures. By introducing scale factor SF, we allow users to simply parameterize T^2K^2 and achieve some scaling, though it could be pushed further in terms of data volume.

In future work, we plan to expand T^2K^2's dataset significantly to aim at big data-scale volume. We also intend to further our proof of concept and validation efforts by benchmarking other NoSQL database systems and gain insight regarding their capabilities and shortcomings. We also plan to adapt T^2K^2 so that it runs in the Hadoop and Spark environments.

References

1. Bringay, S., Béchet, N., Bouillot, F., Poncelet, P., Roche, M., Teisseire, M.: Towards an on-line analysis of tweets processing. In: International Conference on Database and Expert Systems Applications (DEXA). pp. 154–161 (2011)
2. Cooper, J.D., Robinson, M.D., Slansky, J.A., Kiger, N.D.: Literacy: Helping students construct meaning. Cengage Learning (2014)
3. Darmont, J.: Data Processing Benchmarks, pp. 146–152. Encyclopedia of Information Science and Technology (3rd Edition), IGI Global, Hershey, PA, USA (2014)
4. Ferrarons, J., Adhana, M., Colmenares, C., Pietrowska, S., Bentayeb, F., Darmont, J.: PRIMEBALL: a parallel processing framework benchmark for big data applica-

tions in the cloud. In: 5th TPC Technology Conference on Performance Evaluation and Benchmarking (TPCTC 2013). LNCS, vol. 8391, pp. 109–124 (2014)
5. Gattiker, A.E., Gebara, F.H., Hofstee, H.P., Hayes, J.D., Hylick, A.: Big data text-oriented benchmark creation for Hadoop. IBM Journal of Research and Development 57(3/4), 10:1–10:6 (2013)
6. Gray, J.: The Benchmark Handbook for Database and Transaction Systems (2nd Edition). Morgan Kaufmann (1993)
7. Guille, A., Favre, C.: Event detection, tracking, and visualization in twitter: a mention-anomaly-based approach. Social Network Analysis and Mining 5(1), 18 (2015)
8. Kılınç, D., Özçift, A., Bozyigit, F., Yildirim, P., Yücalar, F., Borandag, E.: TTC-3600: A new benchmark dataset for turkish text categorization. Journal of Information Science 43(2), 174–185 (2017)
9. Lewis, D.D., Yang, Y., Rose, T.G., Li, F.: RCV1: A new benchmark collection for text categorization research. Journal of Machine Learning Research 5, 361–397 (2004)
10. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to information retrieval. Cambridge University Press (2008)
11. O'Shea, J., Bandar, Z., Crockett, K.A., McLean, D.: Benchmarking short text semantic similarity. International Journal of Intelligent Information and Database Systems 4(2), 103–120 (2010)
12. Paltoglou, G., Thelwall, M.: A study of information retrieval weighting schemes for sentiment analysis. In: 48th Annual Meeting of the Association for Computational Linguistics. pp. 1386–1395 (2010)
13. Partalas, I., Kosmopoulos, A., Baskiotis, N., Artières, T., Paliouras, G., Gaussier, É., Androutsopoulos, I., Amini, M., Gallinari, P.: LSHTC: A benchmark for large-scale text classification. CoRR abs/1503.08581 (2015)
14. Ravat, F., Teste, O., Tournier, R., Zurfluh, G.: Top_keyword: an aggregation function for textual document OLAP. In: 10th International Conference on Data Warehousing and Knowledge Discovery (DaWaK). pp. 55–64 (2008)
15. Reagan, A.J., Tivnan, B.F., Williams, J.R., Danforth, C.M., Dodds, P.S.: Benchmarking sentiment analysis methods for large-scale texts: A case for using continuum-scored words and word shift graphs. CoRR abs/1512.00531 (2015)
16. Spärck Jones, K., Walker, S., Robertson, S.E.: A probabilistic model of information retrieval: development and comparative experiments: Part 1. Information Processing & Management 36(6), 779 – 808 (2000)
17. Spärck Jones, K., Walker, S., Robertson, S.E.: A probabilistic model of information retrieval: development and comparative experiments: Part 2. Information Processing & Management 36(6), 809 – 840 (2000)
18. Truică, C.O., Darmont, J., Velcin, J.: A scalable document-based architecture for text analysis. In: International Conference on Advanced Data Mining and Applications (ADMA). pp. 481–494 (2016)
19. Wang, L., Dong, X., Zhang, X., Wang, Y., Ju, T., Feng, G.: TextGen: a realistic text data content generation method for modern storage system benchmarks. Frontiers of Information Technology & Electronic Engineering 17(10), 982–993 (2016)