Benchmarking Top-K Keyword and Top-K Document Processing with T^2K^2 and $T^2K^2D^2$

Le calcul des *top-k* mots clés et documents au banc d'essais avec T^2K^2 et $T^2K^2D^2$

Ciprian-Octavian Truică
Université Polytechnique de Bucarest
Roumanie
ciprian.truica@cs.pub.ro

Jérôme Darmont
Université de Lyon, Lyon 2, ERIC EA 3083
France
jerome.darmont@univ-lyon2.fr

Alexandru Boicea
Université Polytechnique de Bucarest
Roumanie
alexandru.boicea@cs.pub.ro

Florin Rădulescu
Université Polytechnique de Bucarest
Roumanie
florin.radulescu@cs.pub.ro

Avec le développement continu et croissant des contenus textuels sur le Web, notamment (mais pas uniquement) dans les médias sociaux, l'analyse de texte est devenue un tendance lourde qui soulève encore aujourd'hui de nombreux problèmes. Parmi ces méthodes d'analyse, la recherche des *k* mots clés ou documents les plus fréquents (*top-k keywords/documents*) est très courante. Par exemple, extraire les *k* termes les plus fréquents d'un corpus permet de déterminer des tendances [5, 14] ou de détecter des événements [10] ; et découvrir les *k* documents les plus similaires à une requête est bien sûr la tâche principale des moteurs de recherche. Par ailleurs, calculer des *top-k* mots clés et documents nécessite un vocabulaire pondéré, qui peut aussi être utilisé dans de nombreux autres types d'analyse de texte comme la recherche de modèles thématiques (*topic modeling*) [2] et le regroupement (*clustering*) [1].

Afin de comparer des combinaisons de méthodes de pondération, des stratégies de calcul, ainsi que l'efficacité de différentes implémentations, les approches par bancs d'essais (*benchmarking*) sont courantes. Toutefois, les bancs d'essais dédiés aux mégadonnées (*big data*) [11, 15, 19] se focalisent essentiellement sur des opérations MapReduce et ne modélisent pas souvent des scénarios basés sur des documents textuels. De plus, les rares qui le font se restreignent au paradigme Hadoop et demeurent à des stades de méthodologie [8] ou de spécifications [7].

Par ailleurs, le calcul de pondérations au niveau de la couche applicative peut se révéler inefficace sur de gros volumes de données, car toute l'information doit être lue et traitée à différents niveaux en plus de celui du stockage. Une meilleure approche consiste à traiter l'information au niveau du stockage, en utilisant des fonctions d'agrégation et en retournant le résultat à la couche applicative.

C'est pourquoi nous avons proposé, dans un premier temps, le banc d'essais T^2K^2 (*Twitter Top-K Keywords Benchmark*), qui s'appuie sur le stockage en base de données d'un corpus de *tweets* et y associe des requêtes de complexité et de sélectivité variées [16]. Nous avons conçu T^2K^2 de manière la plus générique possible. Il permet en effet de comparer différentes méthodes de pondération, des implémentations logiques et physiques de bases de données, ainsi que des plateformes complètes d'analyse de texte, en termes d'efficacité de calcul. Dans cet article, nous poursuivons ce travail par les contributions suivantes [17].

(1) Comme le modèle de données de T^2K^2 est suffisamment générique pour prendre en compte tout type de document textuel (et pas seulement des *tweets*), nous complétons son modèle de charge par des requêtes de calcul de *top-k* documents.

(2) Nous illustrons la pertinence et la généricité de T^2K^2 par la comparaison des méthodes de pondération TF-IDF et Okapi BM25, d'une part, et leur emploi dans des implémentations relationnelles (Oracle, PostgreSQL) et orientée document (MongoDB), d'autre part.

(3) Puisque les requêtes de T^2K^2 sont analytiques par nature, nous faisons l'hypothèse qu'un modèle en étoile d'entrepôt de données peut améliorer les temps de réponses aux requêtes. En conséquence, nous proposons une évolution de T^2K^2 appelée $T^2K^2D^2$, les deux derniers « D » signifiant *Dimensional* et *Documents*, respectivement.

(4) Nous complétons nos premières expériences sur les *top-k* mots clés et documents avec le banc d'essais $T^2K^2D^2$, toujours en comparant les méthodes de pondération TF-IDF et Okapi BM25 sur des implémentations Oracle, PostgreSQL et MongoDB.

Nous avons conçu le modèle de charge de nos bancs d'essais en journalisant le travail de spécialistes en linguistique computationnelle sur une plateforme d'analyse de texte [18] et des données réelles. Après analyse et regroupement (*clustering*) des requêtes similaires, nous avons sélectionné 8 requêtes (4 pour T^2K^2, 4 pour $T^2K^2D^2$) que nous considérons suffisamment génériques pour pouvoir comparer des systèmes similaires d'analyse de texte.

Par ailleurs, il est important de remarquer que nos bancs d'essais ne s'appliquent pas à l'évaluation de systèmes de recherche d'information (RI). Ils ciblent en effet des systèmes utilisant des bases de données et construits comme des surcouches de systèmes de RI, le but étant de faciliter des tâches de fouille de texte (*text mining*) comme la classification, la recherche de modèles thématiques ou la détection d'événements. Les requêtes de nos bancs d'essais ciblent des tâches d'apprentissage automatique (*machine learning*) pour

lesquelles il est important d'analyser différents sous-ensembles d'un jeu de données pour en extraire de la connaissance.

A contrario, les systèmes de RI ne gèrent pas bien les sous-ensembles d'un corpus initial, ni les jeux de données dont la taille varie avec le temps, les poids n'étant pas recalculés. Cela induit deux problèmes de reproductibilité des résultats [12]. Premièrement, l'absence de recalcul des poids induit des erreurs dans les fonctions de classement (*ranking*) utilisées pour le calcul des *top-k* mots clés et documents. Deuxièmement, les poids doivent être recalculés à chaque fois que la taille du corpus change, ce qui provoque de nombreuses opérations d'écriture coûteuses.

L'approche base de données permet de résoudre ces problèmes de reproductibilité en calculant les poids de manière dynamique au moment de la recherche, en utilisant des champs de données (*fielded data*). Bien que les systèmes de RI utilisent aussi des champs de données [6], ils ne calculent les poids qu'une seule fois et les fonctions de classement ne les mettent à jour ni quand des sous-ensembles du corpus sont utilisés, ni quand le volume de donnée croît [3], ce qui provoque des erreurs dans le calcul des *top-k* mots clés et documents.

En revanche, calculer les poids automatiquement est parfaitement adapté à l'analyse de sous-ensembles du corpus initial ou à des corpus en évolution, comme par exemple des flux de données [4]. Les systèmes de gestion de bases de données (SGBD) étant conçus pour gérer de grands volumes de données avec un haut débit et capables de manipuler facilement des volumes de données croissants grâce à des opérations CRUD (*Create, Read, Update, Delete*) optimisées, ils sont plus à même de prendre en charge des jeux de données dont la taille évolue et d'analyser des sous-ensembles du corpus.

Nos résultats expérimentaux sont synthétisés dans le Tableau 1, qui indique le SGBD qui a obtenu les meilleures performances de calcul des *top-k* mots clés et documents en fonction du banc d'essais (en mode mono-serveur ou distribué) et de la méthode de pondération utilisés. Les résultats détaillés sont discutés dans [17].

Table 1: Meilleur système pour les calculs *top-k*

Système de pondération	TF-IDF	Okapi BM25
T^2K^2	MongoDB	Oracle
T^2K^2 distribué	MongoDB	Oracle
$T^2K^2D^2$	Oracle	Oracle
$T^2K^2D^2$ distribué	Oracle	Oracle

Pour terminer, nous avons conçu nos bancs d'essais selon les critères de Jim Gray [9] : pertinence, portabilité, simplicité et scalabilité. Dans nos travaux futurs, nous prévoyons d'étendre de manière significative le jeu de données de T^2K^2 et $T^2K^2D^2$ pour parvenir à un volume à l'échelle des mégadonnées (scalabilité). Nous allons également adapter nos bancs d'essais pour qu'ils s'exécutent dans les environnements distribués tels qu'Hadoop et Spark (portabilité). De plus, nous allons poursuivre nos efforts de preuve de concept et de validation en comparant d'autres SGBD NoSQL que MongoDB (portabilité), afin de déterminer leurs points forts et leurs limites (pertinence). Enfin, nous avons considéré dans cet article que les méthodes de pondération TF-IDF et Okapi BM25 étaient suffisamment représentatives pour des tâches d'apprentissage automatique. Cependant, la prochaine version de nos bancs d'essais devrait en inclure d'autres (pertinence), comme la KL-divergence [13].

RÉFÉRENCES

[1] Charu C. Aggarwal and ChengXiang Zhai. 2012. A Survey of Text Clustering Algorithms. In *Mining Text Data*. Springer, 77–128. https://doi.org/10.1007/978-1-4614-3223-4_4

[2] Rubayyi Alghamdi and Khalid Alfalqi. 2015. A Survey of Topic Modeling in Text Mining. *International Journal of Advanced Computer Science and Applications* 6, 1 (2015), 147–153. https://doi.org/10.14569/IJACSA.2015.060121

[3] P. Bellot, A. Doucet, S. Geva, S. Gurajada, J. Kamps, G. Kazai, M. Koolen, A. Mishra, V. Moriceau, J. Mothe, M. Preminger, E. SanJuan, R. Schenkel, X. Tannier, M. Theobald, M. Trappett, A. Trotman, M. Sanderson, F. Scholer, and Q. Wang. 2013. Report on INEX 2013. *SIGIR Forum* 47, 2 (2013), 21–32. https://doi.org/10.1145/2568388.2568393

[4] Albert Bifet and Eibe Frank. 2010. Sentiment Knowledge Discovery in Twitter Streaming Data. In *Discovery Science*. Springer Berlin Heidelberg, 1–15. https://doi.org/10.1007/978-3-642-16184-1_1

[5] Sandra Bringay, Nicolas Béchet, Flavien Bouillot, Pascal Poncelet, Mathieu Roche, and Maguelonne Teisseire. 2011. Towards an On-Line Analysis of Tweets Processing. In *International Conference on Database and Expert Systems Applications (DEXA)*. 154–161. https://doi.org/10.1007/978-3-642-23091-2_15

[6] Matt Crane, J. Shane Culpepper, Jimmy Lin, Joel Mackenzie, and Andrew Trotman. 2017. A Comparison of Document-at-a-Time and Score-at-a-Time Query Evaluation. In *10th ACM International Conference on Web Search and Data Mining (WSDM)*. ACM, 201–210. https://doi.org/10.1145/3018661.3018726

[7] Jaume Ferrarons, Mulu Adhana, Carlos Colmenares, Sandra Pietrowska, Fadila Bentayeb, and Jérôme Darmont. 2014. PRIMEBALL : a Parallel Processing Framework Benchmark for Big Data Applications in the Cloud, In 5th TPC Technology Conference on Performance Evaluation and Benchmarking (TPC-TC). *LNCS* 839, 109–124. https://doi.org/10.1007/978-3-319-04936-6_8

[8] Anne E. Gattiker, Fadi H. Gebara, H. Peter Hofstee, J. D. Hayes, and A. Hylick. 2013. Big Data text-oriented benchmark creation for Hadoop. *IBM Journal of Research and Development* 57, 3/4 (2013), 10 :1–10 :6. https://doi.org/10.1147/JRD.2013.2240732

[9] Jim Gray. 1993. *The Benchmark Handbook for Database and Transaction Systems (2nd Edition)*. Morgan Kaufmann.

[10] Adrien Guille and Cécile Favre. 2015. Event detection, tracking, and visualization in Twitter : a mention-anomaly-based approach. *Social Network Analysis and Mining* 5, 1 (2015), 18. https://doi.org/10.1007/s13278-015-0258-0

[11] Shengsheng Huang, Jie Huang, Jinquan Dai, Tao Xie, and Bo Huang. 2010. The HiBench benchmark suite : Characterization of the MapReduce-based data analysis. In *Workshops Proceedings of the 26th International Conference on Data Engineering (ICDE)*. 41–51. https://doi.org/10.1109/ICDEW.2010.5452747

[12] Jimmy Lin, Matt Crane, Andrew Trotman, Jamie Callan, Ishan Chattopadhyaya, John Foley, Grant Ingersoll, Craig Macdonald, and Sebastiano Vigna. 2016. Toward Reproducible Baselines : The Open-Source IR Reproducibility Challenge. In *Advances in Information Retrieval*. Springer, 408–420. https://doi.org/10.1007/978-3-319-30671-1_30

[13] Fiana Raiber and Oren Kurland. 2017. Kullback-Leibler Divergence Revisited. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*. ACM, 117–124. https://doi.org/10.1145/3121050.3121062

[14] Franck Ravat, Olivier Teste, Ronan Tournier, and Gilles Zurfluh. 2008. Top_Keyword : an Aggregation Function for Textual Document OLAP. In *10th International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*. 55–64. https://doi.org/10.1007/978-3-540-85836-2_6

[15] Transaction Processing Performance Council. 2016. TPC Express Benchmark HS Standard Specification Version 1.4.2. http://www.tpc.org

[16] Ciprian-Octavian Truică and Jérôme Darmont. 2017. T^2K^2 : The Twitter Top-K Keywords Benchmark, In 21st European Conference on Advances in Databases and Information Systems (ADBIS). *CCIS*, 21–28. https://doi.org/10.1007/978-3-319-67162-8_3

[17] Ciprian-Octavian Truică, Jérôme Darmont, Alexandru Boicea, and Florin Rădulescu. 2018. Benchmarking Top-K Keyword and Top-K Document Processing with T^2K^2 and $T^2K^2D^2$. *Future Generation Computer Systems* 85 (August 2018), 60–75. Special issue on Big Data Benchmarking.

[18] Ciprian-Octavian Truică, Jérôme Darmont, and Julien Velcin. 2016. A Scalable Document-based Architecture for Text Analysis, In International Conference on Advanced Data Mining and Applications (ADMA). *LNAI* 10086, 481–494. https://doi.org/10.1007/978-3-319-49586-6_33

[19] Lei Wang, Jianfeng Zhan, Chunjie Luo, Yuqing Zhu, Qiang Yang, Yongqiang He, Wanling Gao, Zhen Jia, Yingjie Shi, Shujie Zhang, Chen Zheng, Gang Lu, Kent Zhan, Xiaona Li, and Bizhu Qiu. 2014. BigDataBench : A big data benchmark suite from internet services. In *20th IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 488–499. https://doi.org/10.1109/HPCA.2014.6835958