A scalable approach to near real-time sentiment analysis on social networks

G. Amati, S. Angelini, M. Bianchi, L. Costantini, G. Marcone

Fondazione Ugo Bordoni, Viale del Policlinico 147, 00161 Roma, Italy
{gba,sangelini,mbianchi,lcostantini,gmarcone}@fub.it

Abstract. This paper reports about results collected during the development of a scalable Information Retrieval system for near real-time analytics on social networks. More precisely, we present the end-user functionalities provided by the system, we introduce the main architectural components, and we report about performances of our multi-threaded implementation. Since sentiment analysis functionalities are based on techniques for estimating document category proportions, we report about a comparative experimentation aimed to analyse the effectiveness of such techniques.

1 Introduction

The development of platforms for near real-time analytics on social networks poses very challenging research problems to the Artificial Intelligence and Information Retrieval communities. In this context, sentiment analysis is a tricky task. In fact, sentiment analysis for social networks can be defined as a *search-and-classify* task, that is a pipeline of two processes: retrieval and classification [12], [14]. The accuracy of a search-and-classify task thus suffers of the multiplicative effects of independent errors produced by both the retrieval and the classification. The search-and-classify task however is just an example of a most general problem of near real-time analytics. Near real-time analytics is actually based on five main tasks: the retrieval of a preliminary set (the posting lists of the query terms), the assignment of a retrieval score to these documents, the application of binary filters (for example, by selecting documents by period of time and opinion polarity), the mining of hidden entities, and, finally, the final sort to display statistical outcomes and to decorate document pages of results.
All these functionalities must be finally thought and designed to handle *big-data*, as that of Twitter, that generates unbounded streams of data. Moreover, near real-time sentiment analysis for social networks includes end-user functionalities that are typical of either *data-warehouses* or *real-time big data analytics* platforms. For example, the topic of interest is often represented as a large query to be processed in batch mode, and several search tools must support the query specification phase. On the other hand, systems need to continuously index a huge flow of data generated by multiple data-sources, to make new data available as soon as possible, and to prompt reactive detection of incoming events of interest.
In this scenario we report the experience acquired in the development of a system specialized on near-realtime analytics for the Twitter platform.
In Section 2 we describe our system. More precisely, we present end-users functionalities allowing end-users to search, classify and estimate category proportions for real-time analytics. The implementation of these functionalities relies

on some architectural components defined downline of the analysis of a typical retrieval process performed by a search engine. As a consequence, we show how all functionalities can be implemented according to a single retrieval process and how to *scale-up* by a multithreaded parallelization, or *scale-out* by mean of distribution of processes on different computational nodes. We conclude the section reporting the results of an experimentation aimed to assess the performance of our multi-thread implementation. The assessment of the distributed version of the system is still in progress. Even if the system is not yet optimized, the experimentation validates the viability of our solution.
Among all implemented functionalities, in Section 3 we focus on the proportion estimation of categories for sentiment analysis, since quantification for sentiment analysis is particularly complex to be accomplished in near real-time analysis. It is indeed an example of a complex task that requires many steps of Information Retrieval and Machine Learning processing to be performed. Because of this, we describe several techniques for category proportion estimation and we provide their comparison. Section 4 concludes the paper.

2 A scalable system for near real-time sentiment analysis

In order to identify requirements for a system enabling near real-time analysis of phenomena occurring on social networks, we took into consideration two kinds of end-users: *social scientists* and *data scientists*.
Broadly speaking, a social scientist is a user interested in finding answers to questions such as: what are the most relevant/recent tweets, how many tweets convey a positive/negative opinion, what are concepts related to a given topic, how is the trend of a given topic, what are the most important topics, and so on. In general, social scientists interact with the system by submitting several queries formalizing their information needs, they empirically evaluate the quality of the answer provided by the system. The role of social scientist can be played by any user interested in studying or reporting phenomena of social networks that can be connected to scientific discipline such as sociology, psychology, economics, political science, and so on. On the contrast, a data scientist is interested in developing and improving functionalities for social scientists. More precisely, data scientists implement machine learning processes and they take under control the quality of answers provided by the system by means of statistical analyses. Furthermore, they take in charge of define and develop new functionalities for reporting, charting, summarizing, etc.
The following Section presents the end-user functionalities provided by the system. They are the result of a user-requirement analysis activity, jointly conducted by social scientists, data scientist and software engineers.

2.1 End-user functionalities for analytics and sentiment analysis

From the end-user perspective, a system for near real-time analytics and sentiment analysis should provide three main classes of functions: *search*, *count* and *mining* functionalities.
Given a query, *search functionalities* consist in a suite of operations useful to find: the most relevant tweets (*topic retrieval*); the most recent tweets in any interval

of time (*topical timeline*); a representative sample of tweets conveying opinions about the topic (*topical opinion retrieval*); a representative sample of tweets conveying *positive* or *negative* opinions about the topic (*polarity driven topical opinion retrieval*); any mixture of tweets resulting from the combination of relevance, time and opinion search dimensions. Search functionalities are used by social scientist in order to explore tweets indexed by the system, to detect emerging topics, to discover new keywords or accounts to be tracked on Twitter; on other hands, they are used by data scientists to empirically assess the effectiveness of the system.
Count functionalities quantify the result-set size of a given query. As a consequence, they are useful to quantify, for example, the number of positive positive tweets related to a given topic. The system offers two main methods for counting: the *exact count*, that is a database-like function returning the exact number of tweets matching the query, and the *estimated count*, that statistically estimates the number of tweets belonging to a given results-set. As described in Section 3.1 there are some different strategies to perform the estimation count: for sake of exposition we anticipate that the two main approaches are *classify-and-count* and *category size estimation*.
Finally, a suite of *mining functionalities* is available: trending topics, query-related concept mining, geographic distribution of tweets, most representative users for a topic, and so on.
Both count and mining functionalities are mainly used by social scientists for their studying and reporting aims.
In the next Section we show how the above mentioned functionalities can be implemented adopting an Information Retrieval approach.

2.2 A search engine based system architecture

Functionalities presented in the previous Section can be implemented by a system based on a search engine, specifically extended for this purpose. In fact, classic index structures have to be properly configured to host some additional information about tweets. Among the others, an opinion score, a positive opinion score and a negative opinion score, computed at indexing-time and stored in the index, enable the implementation of sentiment analysis functionalities. These scores can be computed by using a dictionary-based approach, as proposed in [1], or by means of an automatic-classifier, such as SVM or Bayesian classifiers. As described in Section 3, these scores can be used at querying-time for implementing functionalities as *exact* and *estimated counting*.
Furthermore, due to the scalability system requirement, index data structures have to support mechanisms for document or term partitioning [13]. In the first case, documents are partitioned into several sub-collections and are separately indexed; in the second case, all documents are indexed as a single collection, and then some data structures (i.e. the lexicon and the posting lists) are partitioned. Even if the term partitioning approach has some advantages in query processing (e.g. making the routing of queries easier and thus resulting in a lower utilization of resources [13]), it does not scale well: because of this we adopt a document partitioning approach.
Once the partitioning approach has been selected, it becomes crucial to define a proper document partition strategy. We opt for partitioning tweets just on the basis of their timestamps: this implies each index contains all tweets generated

during a certain period of time. In our case this strategy is more convenient than others [2],[4],[9],[11],[15], since it is suitable in presence of an unbounded stream of tweets delivered in chronological order; moreover, it enables the optimization of the query process when a time-based constraint is specified for the query.
Finally, we have to decide if to implement a solution to *scale-up* or to *scale-out* in terms of the number of indexed tweets. In the first case, a *multi-index* composed by several *shards* can be created, updated and used on a single machine: as a consequence, the time needed to resolve a query depends on the calculating capacity and the main memory availability on the machine. In the second case, each machine of a computer cluster has to be responsible for a sub-collection and to act as an *indexing* and *query server*: with respect to the time needed to resolve a query, this solution (referred as *distributed index* in the following) exploits the calculating capacity of the entire computer cluster, but introduces some latency due to network communications. Interestingly, in both of the cases, it is possible to define a common set of software components that allow to efficiently implement functionalities presented in Section 2.1. These components, here briefly described, can be implemented to develop an application based on either a multi-index, or a distributed index:

1. *Global Statistics Manager (GSM)*. As soon as new incoming tweets are indexed, the GSM has to update some global statistics, such as the total number of tweets and tokens. Both for multi-index and distributed index solution, the update operation can be simply performed either at query-time, or when the collection changes.
2. *Global Lexicon Manager (GLM)*. The *lexicon* data structure contains the list and statistics of all terms in the collection. Both multi-indexes and distributed indexes require a manager providing information about terms with respect of the entire collection. The GLM can relying on a serialized data structure to be updated every time the collection changes (i.e. a global lexicon), or it can compute at query-time just global information needed to resolve the submitted query.
3. *Score Assigner (SA)*. Any document containing at least one query-term is candidate to be added in the final result-set. SA assigns a ranking score to each document to quantify a relevance degree with respect to the query. Using information provided by GSM and GLM, the scores of document indexed in different shards, or by different query servers, are comparable because computed using global statistics. It is worth noting that opinion scores, needed to sentiment analysis functionalities, are computed once for all at indexing-time, and that they have just to be read in the indexes. In fact, we assume that the classifier model for sentiment analysis does not change over time: as a consequence, any change to global statistics of the collections does not affect already computed sentiment scores, and thus their sentiment classifications.
4. *Global Sorter (S)*. Top-N results are sorted in descending order of score.
5. *Post Processing Retriever (PPR)*: a second pass retrieval can follow the retrieval phase, such as query expansion, or a document score modifier can be applied, such as mixture of relevance, time and sentiment models.
6. *Post Processing Filter and Entity Miner (EM)*: some post-processing operations can be performed in order to filter the final result set by time, country etc. or by sentiment category membership constraints. If the *direct index*, i.e. the posting list of the terms occurring in each document, or other additional

Table 1. Mapping examples of user functionalities over information retrieval processes.

Functionalities	GSM	GLM	SA	S	PPR	EM	D
Query result set count							
Classify and count		X				X	
Category estimation	X	X				X	
Ranking	X	X	X	X			X
Trending topics	X	X			X	X	
Query-related concept mining	X	X	X	X	X	X	

data structures are available, text mining operations can be also applied to the result set, for example: extraction of relevant and trendy concepts, or mentions, or entities related to the query.

7. *Decorator (D)*: once the result set is determined and ordered, some efficient DB-like operations can be eventually performed in order to make results ready for presentation to the final user (e.g. posting records are decorated with metadata such title, timestamp, author, text, etc.).

Table 1 shows which components are involved in the implementation of some exemplifying end-user functionalities. To obtain an efficient implementation of these functionalities it is crucial to design and implement the listed components as more decoupled as possible. It is worth noting the *Query result set count* functionality does not depend on any listed component since it only needs local postings retrieval operations.

2.3 Assessing the performance of a multi-index implementation

We have developed a multi-index based implementation of the system adding new data structure to the Terrier framework[1]. The current version takes advantage of the multi-threading paradigm to parallelize, as much as possible, reading operations from shards.

In order to assess the efficiency of our solution, we use a collection containing more than 153M of tweets, written in English, concerning the FIFA 2014 World Cup (up to half July 2014), and football news (up to half September 2014). Since June 14 to September 14, a new shard has been daily created and added to the multi-index, independently from the number of tweets downloaded in the last 24 hours. The final index contains 76 shards unbalanced in terms of number of contained tweets, as shown in Figure 1 (each shard contains an average of about 2M tweets). We have focused our assessment on the ranking functionality: more precisely, we have used 2127 queries, retrieving an average of about 44,361 tweets each. Table 2 reports the processing time for each component involved in the functionality under testing. In general, observed performances fit our expectation: anyway, we identify a potential bottleneck in the decoration phase. The *decorator* component will have to be carefully developed in the new version of the system based on a distributed index.

[1] http://www.terrier.org

Table 2. Processing time the processing time for each components involved in the functionality. GSM is not reported since it has a negligible processing time. Times are milliseconds averaged on queries. We have run the system on a machine having a quad-core i3 CPU clocked at 3.07 GHz with 8GB RAM. Being the system written in the Java programming language, we have allocated about 4GB to the Java Virtual Machine.

# shards	# docs	GLM	SA	S	D
25	56,304,653	3.76 ms	0.07 ms	7.23 ms	0.05 ms/doc
50	99,912,639	6.86 ms	0.13 ms	9.84 ms	0.06 ms/doc
76	153,137,302	8.20 ms	0.16 ms	10.87 ms	0.07 ms/doc

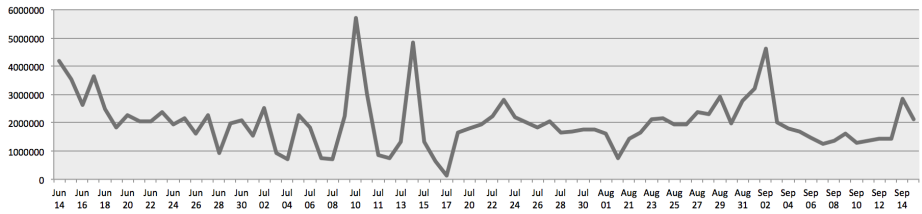

Fig. 1. Number of documents in daily shards used for assessing the performances of the multi-index implementation.

3 Comparing techniques for category proportion estimation

On Twitter, time and sentiment polarity can be important as relevance is for ranking documents. Since sentiment polarity is a classification task, the IR system needs to perform both classification and search tasks in one single shot. In order to obtain a near-real time classification for large data streams, we need to make some computational approximations and to recover the approximation error by introducing a supplementary model able to correct the results, for example by resizing the proportions by estimates of such classification errors [5, 6]. Finally, we correct the number of misclassified items by a linear regression model previously learned on a set of training queries, as presented in [1], or using an adjusted classify & count approach (Section 3.1). At the query time we just combine scores either to aggregate for estimates of retrieval category sizes or to select and sort documents by time, relevance and sentiment polarities.
In this Section we report results of a experimental comparison we conducted on different techniques for category proportion estimation.

3.1 Category proportion estimation

Let $D = \{D_1, \ldots, D_n\}$ be a set of mutually exclusive sentiment categories over the set of tweets Ω, and let q be a topic (story). The problem of size or proportion estimation of sentiment categories for a story consists in specifying the distribution of the categories $P(D_i|q)$ over the result set of the story q.

Such an estimation is similar to that conducted within a typical statistical problem of social sciences, macroeconomics or epidemiological studies. In general if an unbiased sample of the population can be selected, then it can be used to estimate the population categories together with a statistical error due to the size of the sample. For example, Levy & Kass [10] use the *Theorem of Total Probability* on the observed event A to decompose this event over a set of predefined categories. In Information Retrieval, the observed event A can be, for example, the set of the posting lists of a story. We also assume that $\mathrm{P}(A|D_i)$ is obtained by a sample A' of A, that is by $\mathrm{P}(A'|D_i)$. The problem of estimating the category proportions $\mathrm{P}(D_i)$ is determining these probabilities on a sample of observations $A' \subset A$:

$$\mathrm{P}(A') = \sum_{i=1}^{n} \mathrm{P}(A'|D_i)\mathrm{P}(D_i).$$

If we monitor the event A' as aggregated outcome of all observable items in the sample, then we may easily rewrite the Theorem of Total Probability in matrix form as a set of linear equations:

$$\underset{1\times|D|}{\mathrm{P}(A'|D)} \cdot \underset{|D|\times 1}{P(D)}.$$

We simply derive the category proportions $P(D_i)$ by resolving a system of $|D|$ linear equations into $|D|$ variables. From now on we denote all probabilities by $P(\_|q)$ to recall the dependence of observables to the result set of the current query q.

When the assignment of documents of A, or more generally of observables for A, to categories is not performed manually, but automatically, then it is not only the size of the selected sample A that matters, but also both type I and II errors (false positives and false negatives) produced by misclassification that becomes equally significant. In other words, the accuracy of the classifier need also to be known for a correct estimation of all $P(D_i|q)$. If the two types of errors comes out to be similar in size, then the final counting outcomes for category proportions may produce a correct answer. More generally, if the observations is given by a set X of observable variables for the document sample A, then the observables, and their proportions $\mathrm{P}(X|D)$, may be used as a set of training data for a linear classifier to derive $P(D|q)$:

$$\underset{|X|\times 1}{\mathrm{P}(X|q)} = \underset{|X|\times|D|}{\mathrm{P}(X|D,q)} \cdot \underset{|D|\times 1}{P(D|q)}.$$

These equations can be thus resolved, for example, by linear regression. The set of observable variables X can be defined according several approaches.

- *The classify and count methodology*: X is the set of predicted categories $\hat{D}_j$ of a classifier $\hat{D}$. Misclassification errors are given by the conditional probabilities $P(\hat{D}_k|D_j)$ when $k \neq j$. Counting the errors of the classifier in the training data set, and using these measures to correct the category proportions, is at the basis of the adjusted classify and count approach [10, 5, 6].
- *The profile sampling approach*: X is a random subset of word profiles S_j,where a profile is a subset of words occurring in the collection. This approach is at the basis of Hopkins & King's method [7].
- *The cumulative approach*: X is a set of weighted features f_j of a trained classifier (a weighted sentiment dictionary) [1]. The classifier model then can

be used to score each document in the collection. Differently from Hopkins & King's method, that counts occurrences of an unbiased covering set of profiles for a topic, the classifier approach correlates a cumulative category score with category proportions for a topic.

Adjusted-Classify and Count The observations A are obtained by a classifier $\hat{D}$ for the categories D

$$\mathrm{P}(\hat{D}_j|q) = \sum_{i=1}^{n} \mathrm{P}(\hat{D}_j|D_i,q)\mathrm{P}(D_i|q)\ j{=}1,\ldots,\ n.$$

We pool the queries results, that is $\mathrm{P}(\hat{D}'_j|D'_i,q) = \mathrm{P}(\hat{D}'_j|D'_i)$ on a training data set D' and a set of queries. The estimates derive from this pooling set, (i.e.$P(D_i) = P(D'_i)$) solving a simple linear system of $|D|$ equations with $|D|$ variables:

$$\underset{|D|\times 1}{\mathrm{P}(A|q)} = \underset{|D|\times|D|}{\mathrm{P}(A'|D)} \cdot \underset{|D|\times 1}{\mathrm{P}(D|q)}.$$

The methodology is automatic and supervised, and therefore does not need to start over at each query. The accuracy of the classifier does not matter, since the misclassification errors are used for the estimation of category sizes. On the other hand, being not based on a query-by-query learning model, it does not achieve as high precision as with the manual evaluation of Hopkins & King's method.

Hopkins & King's method Let S' be a sample of profiles of words of the vocabulary $\mathbf{V}$, that is $S' \subset S = 2^{\mathbf{V}}$, able to cover *well enough* the space of events, and let A be the set of relevant documents for a topic q. Let us assess the sentiment polarities of a sample A' of A. About 500 evaluated documents will suffice for a statistically significant test. The partition of A' over the categories D will yield the statistics for the occurrences of S' in each category, and these proportions are used to estimate $\mathrm{P}(A|D,q)$. $P(A)$ instead will be estimated by $P(S')$, that is the total number of occurrences of the word profiles of S' in the sample A' with respect to all word profiles occurring in A'.
The category proportions $P(D|q)$ are estimated as the coefficients of the linear regression model

$$\underset{|A|\times 1}{\mathrm{P}(A|q)} = \underset{|A|\times|D|}{\mathrm{P}(A|D,q)} \cdot \underset{|D|\times 1}{\mathrm{P}(D|q)}.$$

This is not a supervised methodology, as it would be with an automated classifier. It is based on counting word profiles from a covering sample. The advantage is a statistically significantly high accuracy (almost 99%, see Table 3). However, there are many drawbacks.The methodology needs to start over at each query, and to achieve such a high accuracy, a long and costly activity of human evaluation of documents is required. The word profile counting is anyway complex since profiles are arbitrary subsets of a very large dictionary, and data are very sparse in Information Retrieval. Moreover, the query-by-query linear regression learning model is also time consuming. In conclusion, this method is not based on a supervised learning model, but it is essentially driven by a manual process, and linear regression and word profiles counting are just used to smooth the maximum likelihood category estimators.

Cumulative approach The cumulative approach is a supervised learning technique that consists in the use of a linear regression to predict and smooth a sentiment category size on the basis of a cumulative score of documents [1]. The approach is information theoretic: for each category, the set F of features for a category are made up of the most informative terms, or equivalently, the highest coding code in that category. Differently from Levy & Kass-Forman's misclassification recovery model, there is not a pipeline of computational processes to perform, namely classifying, then counting, and finally adjusting the category sizes with the number of estimated misclassified items. The technique of the cumulative approach simply correlate the category size with the total number of bits used to code the occurring category features. Since information is additive, the linear regression model is the natural choice that sets up such a correlation over a set of features spanning over a set of training queries. Similarly to the adjusted classify and count approach the precision of this methodology is high and is reported on Section 3.2.

3.2 Experimentation

To assess the effectiveness of the classifier-based quantification, we have build an annotated corpus composed by 6305 tweets manually classified on the basis of the contained opinion. More precisely: 1358 tweets was classified as *positive* (i.e. containing a positive opinion), 2293 as *negative* (i.e. containing a negative opinion), 382 as *mixed* (i.e. containing both positive and negative opinions); 1959 as *neutral* (i.e. not containing opinions), 313 as *not classifiable*.
We have run two sets of experiments. We have first statistical technique to smooth the proportions from a manual document sample assessment. This experiment is essentially manual because requires a training set for each query. For each query instead of the word profiles as used in the proposed by Hopkins & King we have used two standard classifiers (Multinomial Naive Bayes, MNB, and SVM with a linear kernel), and the adjusted classify & count (ACC) as maximum likelihood estimate smoothing technique. However, Hopkins & King's results are hardly reproducible since the set of admissible profiles are generated by a complex feature selection, and also a portion of negative examples are removed from the training set of the query. Indeed, these profiles are generated by an adaptation of the technique by King and Lu [8], that randomly chooses subsets of between approximately 5 and 25 words as admissible profiles. This number of words is determined empirically through cross-validation within the labeled set. Therefore, we show our results in comparison to their method on Table 3 as only reported in their paper [7].
Table 4 shows that the supervised methods with the adjusted classify & count (ACC) technique achieves a very high precision (96.63%-97.86%), i.e. a Mean Absolute Proportion error similar to that of Hopkins & King, with a supervised learning process that is not tailored on a single query only, but trained over a set of about 30 queries and with a 5-fold cross validation. The difference of Mean Absolute Proportion error for 30 queries produced by a search like classification process with respect to Hopkins & King method with a single query, is minimal and not statistically significant.
This first outcomes on Table 4 show that standard supervised classification methods can be effectively applied, and fast implemented, for quantification of sentiment analysis of new queries. The second experiment on Table 5 indeed shows

Table 3. Performance of Hopkins & King Approach (HKA) [7] and Support Vector Machine with linear and polynomial kernels.

Percent of Blog Posts Correctly Classified				
	In-Sample Fit	In-Sample 2-Cross-Validation	Out-of-Sample 2-Cross-Validation	Mean Absolute Proportion Error
HKA	-	-	-	1.2
Linear	67.6	55.2	49.3	7.7
Polynomial	99.7	48.9	47.8	5.3

Table 4. Performance of Classify & Count and Adjusted Classify & Count (ACC) for MNB and SVM Classifiers. Each query has its training data set.

Percent of Tweets Correctly Classified			
	# queries	Out-of-Data-Sample Cross-Validation	Mean Absolute Proportion Error
SVM	30	78.82	2.01
MNB	30	81.12	4.25
ACC-SVM	30	78.82	3.37
ACC-MNB	30	81.12	2.14
HKA	1	-	1.2

that the ACC smoothing with the use of classifiers is a fully automated supervised method that performs highly with new queries as the manual classification of HKA on a single query. The classifiers were trained using a set of about 30 queries and 6-fold cross validation, where each test set has new documents coming from the result sets of the new queries (Out-Query-Sample Cross Validation). We also report the sample fit for each fold (In-Query-Sample Fit Cross-Validation) that shows that an almost perfect category counting with the SVM classifier.
Notice that, the Classify & Count process (CC) is mush less prone to error than the individual classification accuracy, because of possible error type balancing effect (see Table 5). However, there is not a correlation between individual classification accuracy and Mean Absolute Error Rate of the CC process, so that the CC approach cannot ever be considered reliable estimation or statistically significant. Finally, the cumulative approach achieves high effectiveness (Multiple R-squared is 0.9781 for the negative category with 5-fold cross validation on the same set of queries) [1].

4 Conclusion

This paper reported some experiences gained during the development of a scalable system for real-time analytics on social networks.
We have presented how some architectural components resulting from the analysis of a typical querying process that can be used to implement several func-

Table 5. Performance of Classify & Count and Adjusted Classify & Count (ACC) for MNB and SVM Classifiers. Data set is made up of 30 queries, divided in test set and training set of queries.

Percent of Tweets Correctly Classified				
	In-Queries-Sample Fit Cross-Validation	Mean Absolute Proportion Error	Out-of-Queries-Sample Cross-Validation	Mean Absolute Proportion Error
SVM	99.85	0.05	74.76	5.57
MNB	94.26	3.00	78.46	3.95
ACC-SVM	99.85	0.03	74.76	6.23
ACC-MNB	94.26	1.99	78.46	8.91

tionalities of the system. These components can be adopted both for developing a multi-index and a distributed index implementation of the system. We also identified a potential bottleneck in the decoration phase: the related component has to be carefully developed in the distributed version of the system.
Furthermore, we have shown how to estimate real-time document category proportions for topical opinion retrieval for big data. Outcomes are produced either by a direct count or by estimation of category sizes based on a supervised automated classification with a smoothing technique to recover the number of misclassified documents. The use of MNB and SVM classifiers or information-based dictionaries to estimate category proportions are highly effective and achieves almost perfect accuracy if a training phase on the query is also performed.
Search, classify and quantification for analytics can be thus effectively conducted in real-time.

Acknowledgement: Work carried out under the Research Agreement between Almawave and Fondazione Ugo Bordoni.

References

[1] Giambattista Amati, Marco Bianchi, and Giuseppe Marcone. Sentiment estimation on twitter. In *IIR*, pages 39–50, 2014.

[2] C. S. Badue, R. Baeza-Yates, B. Ribeiro-Neto, A. Ziviani, and N. Ziviani. Analyzing imbalance among homogeneous index servers in a web search system. *Inf. Process. Manage.*, 43(3):592–608, 2007.

[3] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern Information Retrieval*, volume 463. ACM press New York, 1999.

[4] Jamie Callan. Distributed Information Retrieval. In *In: Advances in Information Retrieval*, pages 127–150. Kluwer Academic Publishers, 2000.

[5] George Forman. Counting positives accurately despite inaccurate classification. In João Gama, Rui Camacho, Pavel Brazdil, Alípio Jorge, and Luís Torgo, editors, *ECML*, volume 3720 of *Lecture Notes in Computer Science*, pages 564–575. Springer, 2005.

[6] George Forman. Quantifying counts and costs via classification. *Data Min. Knowl. Discov.*, 17(2):164–206, 2008.

[7] Daniel Hopkins and Gary King. A method of automated nonparametric content analysis for social science. *American Journal of Political Science*, 54(1):229–247, 01/2010 2010.

[8] Gary King, Ying Lu, and Kenji Shibuya. Designing verbal autopsy studies. *Population Health Metrics*, 8(1), 2010.

[9] Leah S. Larkey, Margaret E. Connell, and Jamie Callan. Collection Selection and Results Merging with Topically Organized U.S. Patents and TREC Data. In *CIKM 2000*, pages 282–289. ACM Press, 2000.

[10] P S Levy and E H Kass. A three-population model for sequential screening for bacteriuria. *American J. of Epidemiology*, 91(2):148–54, 1970.

[11] Xiaoyong Liu and Bruce W. Croft. Cluster-based retrieval using language models. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on research and development in information retrieval*, pages 186–193, New York, NY, USA, 2004. ACM Press.

[12] Craig Macdonald, Iadh Ounis, and Ian Soboroff. Overview of the TREC 2007 blog track. In Ellen M. Voorhees and Lori P. Buckland, editors, *TREC*, volume Special Publication 500-274. National Institute of Standards and Technology (NIST), 2007.

[13] Alistair Moffat, William Webber, Justin Zobel, and Ricardo B. Yates. A pipelined architecture for distributed text query evaluation. *Inf. Retr.*, 10(3):205–231, June 2007.

[14] Iadh Ounis, Maarten de Rijke, Craig Macdonald, Gilad Mishne, and Ian Soboroff. Overview of the trec-2006 blog track. In *Text Retrieval Conference*, 2006.

[15] Diego Puppin, Fabrizio Silvestri, and Domenico Laforenza. Query-driven document partitioning and collection selection. In *InfoScale '06: Proceedings of the 1st international conference on Scalable information systems*. ACM Press, 2006.