2023 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)

Towards an adaptive algorithms selection on predicting the update interval for social media feeds

Max-Emanuel Keller, Alexander Döschl, Peter Mandl
HM Hochschule München University of Applied Sciences
Munich, Germany
{max-emanuel.keller, alexander.doeschl, peter.mandl}@hm.edu

Alexander Schill
TU Dresden
Dresden, Germany
alexander.schill@tu-dresden.de

***Abstract*—Efficiently synchronizing data with external sources, such as social media feeds, while minimizing well-timed requests is a challenge in various domains. This research investigates prediction algorithms for determining appropriate update intervals for Facebook and Twitter feeds, considering metrics such as the delay (time between a post's publication and retrieval) and requests per post. Due to variations in update intervals, different algorithms yield diverse results. Selecting the most suitable algorithm for each feed is a time-consuming but crucial task for achieving optimal resource usage. We propose three strategies for algorithm selection: baseline (using a single algorithm per feed), optimum (calculating the best algorithm for each feed), and classification (identifying algorithms through classification). Real-world data from Facebook and Twitter are used to evaluate the strategies, comprehensively assessing their strengths and weaknesses. Findings demonstrate that the strategy optimum identifies the best algorithms, while the strategy classification selects fairly good algorithms at significantly reduced computational effort.**

I. Introduction

Over recent years, social media platforms have attracted many new users. The platforms consequently gained considerable importance from private as well as commercial and scientific perspectives. This leads to real-world and science applications, that rely on data from these platforms. Many of these applications require posts from the feeds of a large number of pages or user profiles. In addition, new posts should be retrieved with as little delay as possible.

In this paper, we study two of the most important platforms, Facebook and Twitter, which provide APIs that can be used to retrieve data. The APIs, however, have a rate limit, so that each endpoint can only be called a certain number of times within a given period of time. Refresh for updates also requires polling of each feed, while only a limited set of feeds can be refreshed on a regular basis. A push-based asynchronous notification on updates with webhooks, invoked by the platforms, would be the obvious solution. That is known by the platform providers why the use of webhooks, albeit possible for both platforms, is however limited to page owners on Facebook or paying users on Twitter. Hence, API users have to deal with the limits.

Aware of the issues, Facebook and Twitter suggest caching the responses of API calls and only re-invoke the API when required. Both, however, do not provide concrete strategies that go beyond traditional web caching. To meet our goal of retrieving new posts with as little delay as possible, prediction algorithms are needed to predict when the next post on a feed will occur in order to retrieve it at the appropriate time.

The feeds, however, can differ greatly in their update intervals, so that various existing prediction algorithms lead to different results. Hence, in this work, we propose an efficient approach for selecting the most suitable prediction algorithms for new posts on feeds of social media platforms.

ASONAM '23, November 6–9, 2023, Kusadasi, Turkey

ACM ISBN 979-8-4007-0409-3/23/11…$15.00
http://dx.doi.org/10.1145/3625007.3627474

II. Problem

We aim to predict the update intervals for feeds of posts. The problem can be described as follows: Let $f_k \in F = \{f_0, \ldots, f_q\}$ be the feed of a Facebook page or Twitter user that contains n_k posts $p_{k,i}$. For simplicity, we refer to $p_{k,i}$ as p_i and n_k as n for an arbitrary feed f_k in the following unless otherwise noted. Each post p_i has a timestamp t_i that denotes the publication time of p_i, as illustrated by Figure 1. The duration between the times of subsequent posts $t_i, t_{i+1};\ t_i < t_{i+1}$, can be defined as real update interval $u_{real} = t_{i+1} - t_i$, also called inter-arrival time. Request $\tau_j \in \{\tau_0, \ldots, \tau_l\}$ denotes the times the feed is requested and the duration between subsequent requests τ_j, τ_{j+1}, can be defined as predicted update interval $u = \tau_{j+1} - \tau_j$. When we request the feed at τ_j posts published at times $t_i < \tau_j$ are retrieved with a delay $d = \tau_j - t_i$ with up to n posts. In some cases there is a window w which limits the amount of posts returned to the interval $(t_{n-w}, t_n];\ n \geq w$.

We aim to calculate an optimal interval u with $d \to 0$. In doing so, we encounter several challenges. First, the intervals between the times of three consecutive posts of the same feed t_i, t_{i+1}, t_{i+2} might vary, with $t_{i+1} - t_i \neq t_{i+2} - t_{i+1}$. Second, different feeds can have completely divergent update intervals. Hence, the interval should be calculated separately for each feed and recalculated at a regular frequency.

In this work we aim to select for each feed $f_k \in F = \{f_0, \ldots, f_q\}$ an individual algorithm a_k from a given set of algorithms $\Omega = \{\omega_a, \omega_b, \ldots\}$. The individual algorithms of

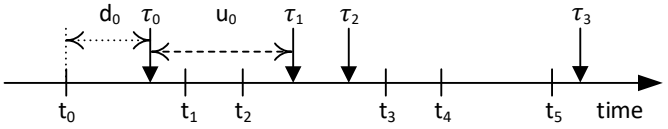

Fig. 1. Posts of a single feed

two feeds f_k, f_{k+1} can be identical ($a_k = a_{k+1}$) or differ ($a_k \neq a_{k+1}$). The entire set of individual algorithms a_k for $f_k \in F$ form the set $A = \{a_0, \ldots, a_q\}$. In order to determine the set of individual algorithms, we select among different temporary sets of algorithms $\Omega' = \{\omega_0, \ldots, \omega_q\}$.

As we will show in Section VI, there is no single algorithm $\omega \in \Omega$ that performs best when applied on every single feed $f_k \in F$. Rather, the best algorithm depends on the particular feed. Hence, we propose *three strategies* for selecting the appropriate algorithm for individual feeds in Section IV.

III. Details

The problem of update interval computation arises from the need to make efficient use of the API calls available, since these are restricted by the rate limits. In this section, we propose metrics for the evaluation and give an overview of several algorithms for the update interval calculation.

A. Metrics

In the following section, we present several algorithms $\omega \in \Omega$ we want to evaluate for the update interval calculation on how they meet our requirements by using appropriate metrics.

The first obvious metric is *delay* D, as the delay between publication and retrieval of a post [1–4]. To apply it for the algorithm comparison, we calculate it with an algorithm ω for every single post p_i with Equation (1), but also as aggregation for all posts of a feed f_k with Equation (2).

$$D(\omega, p_i) = \tau_j - t_i \text{ with } t_i \in [\tau_{j-1}, \tau_j] \tag{1}$$

$$\widetilde{D}(\omega, f_k) = median(D(\omega, p_0), \ldots, D(\omega, p_n)) \tag{2}$$

The delay tends to decrease with an increase in requests. However, a simple increase in requests is in contrast with the rate limit. Hence, we introduce a second metric called *requests per element* RPE, which considers the number of requests. RPE is defined as the number of requests that were executed until at least one new element was found [1, 4]. The $RPE(\omega, \tau_j)$ is undefined if not at least one new post p_i is found. Otherwise, $RPE(\omega, \tau_j)$ is defined as the number of requests $|\tau_y|$ that were made since the last request τ_x where a new post was found, divided by the number of new posts $|p_i|$ with times t_i in $\tau_x \leq t_i < \tau_j$. We calculate RPE for a single request with Equation (3) or for a feed with Equation (4). Given the example in Figure 1, $RPE(\tau_0) = 1$, $RPE(\tau_1) = \frac{1}{2}$, $RPE(\tau_2) =$ undefined, and $RPE(\tau_3) = \frac{2}{3}$. $RPE = 1$ is optimal for a single post at a time, while for more posts, published at the same time, it is $\frac{1}{n}$ for n posts.

$$RPE(\omega, \tau_j) = \begin{cases} \text{undefined} & \text{for } |\tau_x \leq t_i < \tau_j| = 0 \\ \frac{|\tau_x < \tau_y \leq \tau_j|}{|\tau_x \leq t_i < \tau_j|} & \text{for } |\tau_x \leq t_i < \tau_j| > 0 \end{cases} \tag{3}$$

$$\widetilde{RPE}(\omega, f_k) = median(RPE(\omega, \tau_0), \ldots, RPE(\omega, \tau_l)) \tag{4}$$

Both metrics are calculated either for a single feed f_k or for a set of feeds F, using either of two modes, *Feeds* or *Elements*. Mode *Feeds* weights each feed equally, while *Elements* has the effect that feeds with many posts dominate the result. For a set of feeds F, $\widetilde{D}_F$ for mode *Feeds* first calculates a median post delay per feed, before it calculates a second median over it with Equation (5). $\widetilde{D}_E$ for mode *Elements* is calculated as median delay over all posts $p_{k,i}$ of all feeds $f_k \in F = \{f_0, \ldots, f_q\}$ with Equation (6). For a single feed f_k, however, both modes lead to the same result, with $\widetilde{D}_F(\omega, f_k) = \widetilde{D}_E(\omega, f_k)$ in Equation (7). $\widetilde{RPE}_F$ and $\widetilde{RPE}_E$ are calculated accordingly, where undefined values are omitted.

$$\widetilde{D}_F(\Omega', F) = median(\widetilde{D}(\omega_0, f_0), \ldots, \widetilde{D}(\omega_q, f_q)) \tag{5}$$

$$\widetilde{D}_E(\Omega', F) = median \begin{pmatrix} D(\omega_0, p_{0,0}), \ldots, D(\omega_0, p_{0,n_0}), \\ \ldots \\ D(\omega_q, p_{q,0}), \ldots, D(\omega_q, p_{q,n_q}) \end{pmatrix} \tag{6}$$

$$\widetilde{D}_F(\omega, f_k) = \widetilde{D}_E(\omega, f_k) = \widetilde{D}(\omega, f_k) \tag{7}$$

To apply the metrics to different algorithms and make the resulting values comparable, the metrics are normalized as $\widehat{D}_F$, $\widehat{D}_E$, $\widehat{RPE}_F$, $\widehat{RPE}_E$. To do this, for a feed f_k and an algorithm ω, the algorithm $x \in \Omega$ with the best value for the metric, in case of f_k, is set to 1 as the reference value. With Equation (8) and $\widetilde{D}_F$ or $\widetilde{D}_E$ as $\widetilde{D}$, all values are set in relation to the best value, that equals to the minimum. This transforms the values to the interval $(0; 1]$. For a set of Feeds F and a set of algorithms Ω', the set of algorithms $X \in \Omega$ with the best value for the metric, in case of F, is set to 1 as the reference value, in Equation (9). $\widehat{RPE}_F$ and $\widehat{RPE}_E$ are calculated accordingly.

$$\widehat{D}(\omega, f_k) = \frac{\text{arg min}_{x \in \Omega}(\widetilde{D}(x, f_k))}{\widetilde{D}(\omega, f_k)}; \quad \omega \in \Omega; \widetilde{D}(\omega, f_k) > 0 \tag{8}$$

$$\widehat{D}(\Omega', F) = \frac{\text{arg min}_{X \in \Omega}(\widetilde{D}(X, F))}{\widetilde{D}(\Omega', F)}; \quad \Omega' \in \Omega; \widetilde{D}(\Omega', F) > 0 \tag{9}$$

To rank the algorithms for f_k or sets of algorithms for F, all algorithms or sets of algorithms have to be compared twice for $\widehat{D}_F$ and $\widehat{RPE}_F$ or $\widehat{D}_E$ and $\widehat{RPE}_E$. To simplify this,

we transform both metrics into a combined metric *quality* Q. Q is the geometric mean of $\widehat{D}$ and $\widehat{RPE}$, as shown for a feed f_k in Equation (10) and a set of Feeds F in Equation (11). Due to comparability, we also normalize Q, where the algorithm $x \in \Omega$ with the best value for the metric, in case of f_k, that equals the maximum, is set to 1 as the reference value. With Equation (12) all values are set in relation to the best value. Accordingly, for a set of Feeds F and a set of algorithms Ω', the best set of algorithms X, in case of F, is the reference value in Equation (13). For the Equations (10) to (13), $Q, \widehat{D}$, and $\widehat{RPE}$ stand in place for $Q_F, \widehat{D}_F$, and $\widehat{RPE}_F$ or $Q_E, \widehat{D}_E$, and $\widehat{RPE}_E$ accordingly.

$$Q(\omega, f_k) = \sqrt{\widehat{D}(\omega, f_k) \cdot \widehat{RPE}(\omega, f_k)} \tag{10}$$

$$Q(\Omega', F) = \sqrt{\widehat{D}(\Omega', F) \cdot \widehat{RPE}(\Omega', F)} \tag{11}$$

$$\widehat{Q}(\omega, f_k) = \frac{Q(\omega, f_k)}{\text{arg max}_{x \in \Omega}(Q(x, f_k))}; \quad \omega \in \Omega; \arg\max_{x \in \Omega}(Q(x, f_k)) > 0 \tag{12}$$

$$\widehat{Q}(\Omega', F) = \frac{Q(\Omega', F)}{\text{arg max}_{X \in \Omega}(Q(X, F))}; \quad \Omega' \in \Omega; \arg\max_{X \in \Omega}(Q(X, F)) > 0 \tag{13}$$

B. Prediction algorithms and boundaries

We used a previous selection of prediction algorithms [5] and evaluated each one with a range of parameters first, to find the best parameter combination. For a detailed description we refer to [5]. The selected algorithms belong to three categories:

- *Static algorithms*: Static with 1 h, 24 h and 7 d; Static average; MAVSync [6]
- *Adaptive algorithms*: Adaptive TTL [7] with $M \in \{0.1, 1.2\}$; LRU [8] with $k = 6$
- *Poisson algorithms*: IndHist [1] with $\theta = 0.6$; IndHist/TTL [1] with $\theta = 0.6$, $T_{burst} = 2$, $M \in \{0.3, 2.5\}$, $W = 1$ h

The update interval calculation can lead to problems for feeds where the interval u_{real} is very long or extremely short. Hence, we cut the update intervals u calculated by the algorithms within two boundaries [9], illustrated in Equation (14).

$$u_{new} = \min(\beta, \max(\alpha, u)); \quad \alpha = 10 \text{ min}, \beta = 7 \text{ d} \tag{14}$$

IV. Approach

In this section, we propose three strategies for selecting an individual algorithm a_k for each feed $f \in F$. The *strategy Baseline* represents the most straightforward process to select a set of algorithms for a set of feeds, serving as a good base for comparison with more complex strategies. First, each algorithm $\omega \in \Omega$ is applied to all feeds $f_k \in F$ for prediction. Second, we calculate an overall normalized quality ($\widehat{Q}_F(\omega, F)$ and $\widehat{Q}_E(\omega, F)$) with Equations (11) and (13) for each algorithm $\omega \in \Omega$ and F, as well as both modes. The algorithm $x \in \Omega$ providing the highest overall normalized quality is used as the individual algorithm a_k for all feeds $f_k \in F$, resulting in $x = a_k \forall a_k \in A = \{a_0, \ldots, a_q\}$. Regarding the two modes, we further distinguish between *strategy Baseline Feeds* and *Baseline Elements*.

The *strategy Optimum* aims to improve the selection over *strategy Baseline* by selecting the best individual algorithm for each feed $f_k \in F$. Again, each algorithm $\omega \in \Omega$ is applied to all feeds $f_k \in F$, but in contrast to strategy Baseline, a normalized single quality $\widehat{Q}_F(\omega, f_k)$ and $\widehat{Q}_E(\omega, f_k)$ from Equations (10) and (12) is calculated for each combination of algorithm $\omega \in \Omega$ and feed f_k for both modes. The algorithm $x \in \Omega$ for a feed f_k, with the highest normalized single quality, is used as the individual algorithm a_k for f_k. This results in a set of algorithms $a_k \in A = \{a_0, \ldots, a_q\}$.

The *strategy Optimum* reliably selects the best individual algorithm a_k for each feed f_k but also leads to significant costs with $m \times n$ permutations, given m algorithms and n feeds. In contrast, with *strategy Classification*, the individual algorithm a_k for each feed is determined more cost-efficiently, using its properties in two processes *model creation* and *model application*, presented in Figure 2. Both processes depend on a dataset of feeds, which is divided into a *base dataset* and a *new feeds dataset*. The base dataset is used to create the model, while the new feeds dataset contains the remaining feeds an algorithm has to be selected for. The *model creation* utilizes the feeds and their posts from the base dataset. With step 1a, we compute predictions using all algorithms $\omega \in \Omega$ for each feed $f \in F$ of the base dataset. From the predictions, the algorithm x is determined as an individual algorithm a_k for each feed based on the highest normalized single quality from Equations (10) and (12) and stored as a label. In step 2a, features from the feeds of the base dataset are obtained, including temporal and content attributes from the posts of the feed. The *posting pattern* describes the distribution of posts by day of the week, as relative frequencies, over the 168 values of a week (24 h × 7 d), the 24 values of a day (24 h), and the 60 values of an hour (60 min.). The *post interval* contains descriptive statistics on the real update interval u_{real} between posts, including the arithmetic mean, standard deviation, variance, number of intervals, range, kurtosis, skew, percentiles (0 to 100) and a histogram of cumulative relative frequency. The *change history* describes the posts per week (last 52), day (last 30) and the standard deviation of u for the posts of a week (last 52). The *content pattern* includes the concatenated post texts. The features and selected algorithm of each feed are divided by 60 % into a training set and 40 % into a test set. During step 3a, the training of the classifier is performed using the features and labels from the training set. For evaluation, in step 4a, for each feed from the test set, an algorithm is predicted using the classifier, which is finally compared with the actual best algorithm for the given feed. The *model application* uses the feeds from the new feeds dataset, for which only the features need to be calculated in step 1b. In step 2b, an algorithm for each new feed is predicted using the features from step 1b.

V. Related work

The work of [4] aimed to predict new entries in web feeds, finding that the algorithms performance depends on individual update-pattern, why they propose manual algorithm selection. The authors of [10] addressed a similar issue for social media feed (Sina Weibo) before. They assign each feed to one out of four predefined patterns and suggest to choose the prediction algorithm according to it. However, with either approach, the patterns are limited to a finite set, which has to be defined in advance. Unknown patterns or changes over time are not taken into account. Therefore, an alternative to this can be finding unknown patterns through unsupervised learning, e.g., clustering of patterns with k-means for web feeds [11] or with LDA for feeds on social media (Tencent QQ) [12].

VI. Evaluation

In this section, we evaluate the strategies from Section IV using the algorithms and metrics from Section III. We run the evaluation over a dataset with 13,000 feeds from Facebook and Twitter respectively. Each feed of the data set has at least 50 posts during the time period between 2020-01-01 and 2020-12-31, which we divided into a train- and test period. For algorithms that require a training phase (e.g., IndHist), we selected the first 30 posts of each feed to train with; otherwise, these posts were ignored. The approach of strategy classification can be used as described in Figure 2 in practice, but has been slightly modified in this paper. The 13,000 feeds are split 60 % (7,800) into the *base dataset* and 40 % (5,200) into the *new feeds dataset*, with re-use of the new feeds dataset as testset.

To identify the best algorithm for individual feeds with strategy Optimum, we calculated the normalized single qualities $\widehat{Q}(\omega, f)$ from Equation (12) for the 5,200 feeds from the *new feeds dataset* and counted the occurrences of each algorithm as the best. The results show that there is no single algorithm that leads to the best results when applied to each feed $f_k \in F$. Following the *creation of the model* of *strategy Classification* from Figure 2, we trained classifiers with Random Forest and XGBoost. The Figures 7 and 10 illustrate distribution of single qualities of the algorithms selected for the feeds of the new feeds dataset. As the distribution reveals, the single qualities of the algorithms selected with Random Forest (RF) and XGBoost (XGB) exhibit much higher values compared to all single algorithms, with a first quartile (Q1) of 0.73 (Facebook) and 0.83 (Twitter), a median of 0.9 (Facebook) and 0.95 (Twitter), and a third quartile (Q3) of 1.0 each.

Figures 3 and 4 show the normalized quality $\widehat{Q}$ from Equation (13). The strategy Optimum shows the best quality of the strategies for Facebook in both modes and for Twitter in the mode Elements, which also outperforms all algorithms. The strategy Classification achieves very high values for Facebook in both modes, ranking second behind the strategy Optimum in both modes, but performing significantly better than strategies Baseline Feeds and Baseline Elements. On Twitter, the strategy Classification also ranks second in mode Elements, slightly behind strategy Optimum, while even outperforming it in mode Feeds. This is explained by the different calculations of the metrics, which weight the modes Feeds and Elements differently. The strategy Optimum generally achieves a larger number of better values, which are however distributed on fewer feeds in the median, hence the strategy Classification performs better in mode Feeds. This becomes clear by comparing metric D for strategy Optimum and strategy Classification, with focus on the median in the modes Feeds and Elements.

VII. Conclusion and future work

In this paper, we proposed three strategies for selecting algorithms for predicting new posts on social media platforms, with the aim of minimal delays between a post's publication and retrieval. We applied the strategies to 13,000 feeds each from Facebook and Twitter. The results revealed up- and downsides of the strategies. Hence, there is a trade-off between delay and requests per element, imposed by the rate limit. The quality $\widehat{Q}_F, \widehat{Q}_E$ can be good indicators for applicability on individual applications. In a future work, we will analyze the runtime of the strategies Baseline, Optimum and Classification.

References

[1] L. Bright, A. Gal, and L. Raschid, "Adaptive pull-based policies for wide area data delivery," *ACM Transactions on Database Systems*, vol. 31, no. 2, pp. 631–671, 2006.

[2] K. C. Sia, J. Cho, and H.-K. Cho, "Efficient monitoring algorithm for fast news alerts," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 7, pp. 950–961, 2007.

[3] M. Yang, H. Wang, L. Lim, and M. Wang, "Optimizing content freshness of relations extracted from the web using keyword search," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, 2010, p. 819.

[4] S. Reichert, "Analyse und vorhersage der aktualisierungen von web-feeds," Dissertation, TU Dresden, Dresden, 2012.

[5] M.-E. Keller, A. Döschl, P. Mandl, and A. Schill, "When she posts next? a comparison of refresh strategies for online social networks," in *The 23rd International Conference on Information Integration and Web Intelligence*. ACM, 2021, pp. 123–129.

[6] D. Urbansky, S. Reichert, K. Muthmann, D. Schuster, and A. Schill, "An optimized web feed aggregation approach for generic feed types," 2011.

[7] J. Gwertzman and M. Seltzer, "World-wide web cache consistency," in *Proceedings of the 1996 Annual Conference on USENIX Annual Technical Conference*, ser. ATEC '96. USENIX Association, 1996, p. 12.

[8] E. J. O'Neil, P. E. O'Neil, and G. Weikum, "The lru-k page replacement algorithm for database disk buffering," in *Proceedings of the 1993 ACM SIGMOD international conference on Management of data - SIGMOD '93*, 1993, pp. 297–306.

[9] S. Reichert, D. Urbansky, K. Muthmann, P. Katz, M. Wauer, and A. Schill, "Feeding the world," in *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*, 2011, p. 44.

[10] R. Guo, H. Wang, K. Li, J. Li, and H. Gao, "Cuvim: Extracting fresh information from social network," in *Web-Age Information Management*, ser. Lecture Notes in Computer Science / Information Systems and Applications, Incl. Internet/Web, and HCI, 2013, vol. 7923, pp. 351–362.

[11] K. C. Sia, J. Cho, K. Hino, Y. Chi, S. Zhu, and B. L. Tseng, "Monitoring rss feeds based on user browsing pattern," in *Proceedings of the First International Conference on Weblogs and Social Media, ICWSM 2007, Boulder, Colorado, USA, March 26-28, 2007*, 2007.

[12] Q. F. Ying, D. M. Chiu, S. Venkatramanan, and X. Zhang, "Profiling osn users based on temporal posting patterns," in *Companion of the The Web Conference 2018 on The Web Conference 2018 - WWW '18*, P.-A. Champin, F. Gandon, M. Lalmas, and P. G. Ipeirotis, Eds., 2018, pp. 1451–1456.

Notes: Static 7d was omitted in Figures 5 and 8 due to the range of the values, so was Static 1h in Figures 6 and 9.

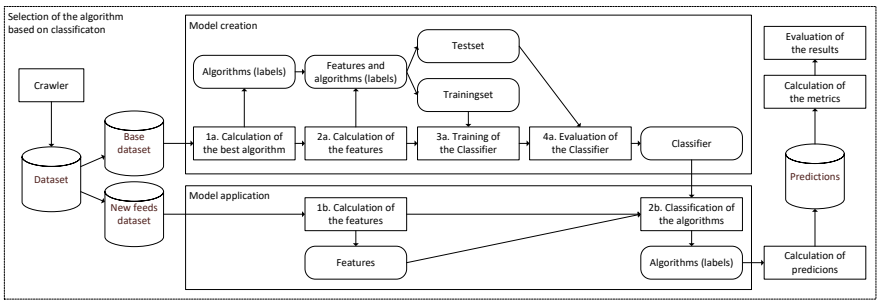

Fig. 2. Strategy Classification

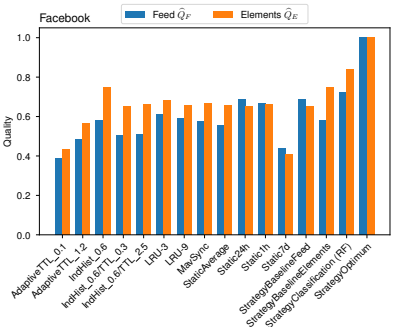

Fig. 3. Facebook - Quality

TABLE I
CLASSIFICATION FACEBOOK

	Precision		Recall		F1-Score		Support
	RF	XGB	RF	XGB	RF	XGB	
AdaptiveTTL_0.1	0.25	0.23	0.26	0.24	0.26	0.24	62
AdaptiveTTL_1.2	0.25	0.00	0.05	0.00	0.09	0.00	19
IndHist_0.6	0.40	0.42	0.60	0.51	0.48	0.46	986
IndHist_0.6/TTL_0.3	0.17	0.07	0.00	0.02	0.01	0.03	236
IndHist_0.6/TTL_2.5	0.26	0.23	0.02	0.11	0.04	0.15	477
LRU-3	0.25	0.16	0.04	0.08	0.08	0.11	426
LRU-9	0.00	0.09	0.00	0.02	0.00	0.03	200
MavSync	0.33	0.20	0.01	0.02	0.03	0.04	142
Static1h	0.39	0.37	0.27	0.39	0.32	0.38	1067
Static24h	0.38	0.39	0.76	0.63	0.50	0.48	1369
Static7d	0.20	0.22	0.05	0.07	0.08	0.10	60
StaticAverage	0.00	0.09	0.00	0.01	0.00	0.01	156
Accuracy					0.38	0.37	5200
Macro avg.	0.24	0.21	0.17	0.17	0.16	0.17	5200
Weighted avg.	0.32	0.31	0.38	0.37	0.30	0.32	5200

TABLE II
CLASSIFICATION TWITTER

	Precision		Recall		F1-Score		Support
	RF	XGB	RF	XGB	RF	XGB	
AdaptiveTTL_0.1	0.43	0.46	0.66	0.61	0.52	0.53	453
AdaptiveTTL_1.2	0.00	0.00	0.00	0.00	0.00	0.00	26
IndHist_0.6	0.41	0.42	0.61	0.52	0.49	0.47	1024
IndHist_0.6/TTL_0.3	0.40	0.36	0.10	0.13	0.15	0.19	231
IndHist_0.6/TTL_2.5	0.28	0.25	0.10	0.18	0.14	0.21	648
LRU-3	0.18	0.16	0.04	0.07	0.07	0.10	378
LRU-9	0.40	0.18	0.01	0.05	0.03	0.08	281
MavSync	0.44	0.70	0.16	0.10	0.24	0.18	68
Static1h	0.39	0.41	0.53	0.54	0.45	0.47	1203
Static24h	0.39	0.36	0.49	0.45	0.44	0.40	748
Static7d	1.00	0.00	0.03	0.00	0.07	0.00	29
StaticAverage	0.00	0.06	0.00	0.01	0.00	0.02	111
Accuracy					0.39	0.38	5200
Macro avg.	0.36	0.28	0.23	0.22	0.22	0.22	5200
Weighted avg.	0.36	0.35	0.39	0.38	0.34	0.35	5200

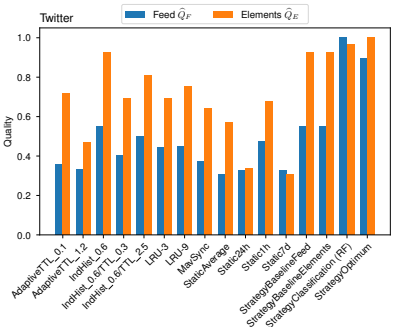

Fig. 4. Twitter - Quality

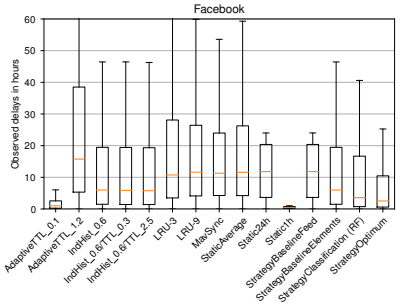

Fig. 5. Facebook - Delay (Elements)

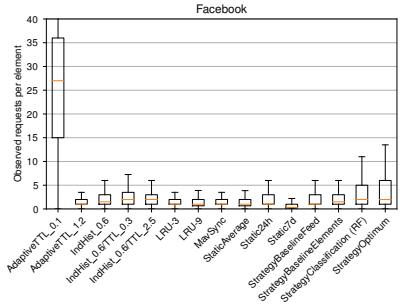

Fig. 6. Facebook - RPE (Elements)

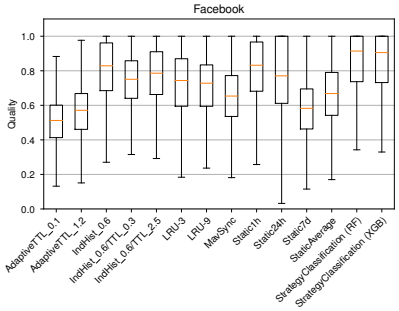

Fig. 7. Facebook - Single qualities (Distrib.)

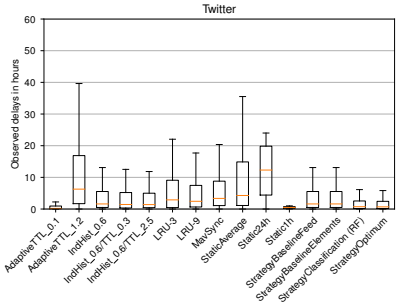

Fig. 8. Twitter - Delay (Elements)

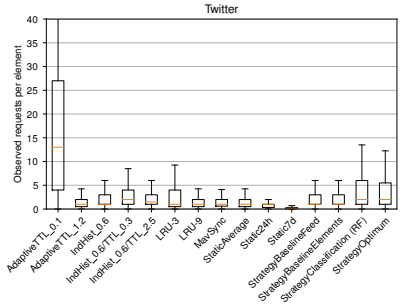

Fig. 9. Twitter - RPE (Elements)

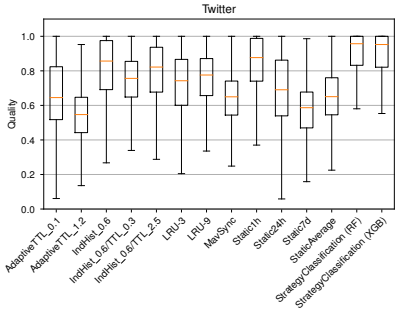

Fig. 10. Twitter - Single qualities (Distrib.)