A Language and an Inference Engine for Twitter Filtering Rules

Alberto Bartoli*, Barbara Carminati†, Elena Ferrari†, Eric Medvet*
*Dipartimento di Ingegneria e Architettura, Università degli Studi di Trieste, Italy
†Dipartimento di Scienze Teoriche e Applicate, Università degli Studi dell'Insubria, Italy

***Abstract*—We consider the problem of the filtering of Twitter posts, that is, the hiding of those posts which the user prefers not to visualize on his/her timeline. We define a language for specifying filtering policies suitable for Twitter posts. The language allows each user to decide which posts to filter out based on his/her sensibility and preferences. Since average users may not have the skills necessary to translate their filtering needs into a set of rules, we also propose a method for inferring a policy automatically, based solely on examples of the desired filtering behavior. The method is based on an evolutionary approach driven by a multi-objective optimization scheme. We assess our proposal experimentally on a real Twitter dataset and the results are highly promising.**

I. Introduction

Filtering of short text messages is becoming more and more relevant in these years, where billion of users use online social networking services to communicate, share and disseminate a considerable amount of information. Despite the many benefits that being exposed to this huge amount of information can bring, there are also many serious shortcomings. One is that of being exposed to the so called "information overload"[1]. Microblogging services are one of the main sources of information overload [1]. For instance, in [2] it is shown that two thirds of Twitter users have felt that they receive too many posts, and over half of Twitter users have felt the need for a tool to filter out the irrelevant ones. There are other further reasons for the need of filtering tools. As an example, a user may want to filter out from his/her timeline messages with disturbing content (e.g., vulgar), or messages about specific sensitive topics (e.g., religion, politics). Social services providers are now trying to cope with this issue. For instance, Twitter has recently announced that it will start showing a selection of tweets that a user who has been away from the service might want to see.

To cope with this issue, in this paper, we first propose a language for expressing filtering rules on Twitter. We believe that, like privacy preferences, also filtering rules are very subjective and may depend on many different dimensions (e.g., the tweet contents, the author of the tweet, etc.). However, the availability of an expressive language, like the one we propose in this paper, is just one of the key building blocks of an effective filtering tool. Indeed, average Twitter users will not have the skills necessary to translate their filtering needs into a set of rules [3]. Moreover, they can also not be fully aware of their filtering needs. Therefore, we complement our language with a tool able to infer filtering rules automatically, from selected examples of the required filtering behaviour. At this purpose, we exploit an evolutionary approach in which individuals representing filtering rules are evolved according to a multi-objective optimization scheme; rules are then combined together by means of a separate-and-conquer strategy which allows to automatically partition examples in suitable subsets. In general, Evolutionary Computation (EC) fits the rules inference task, since it is able to handle examples which are not fully consistent, as it might happen when examples are provided by unskilled users. Indeed, many applications of EC to rules inference have been proposed, some of them concerning security and access control [4], [5], [6], [7].

In the paper, besides presenting our approach, we show the results of a first experimental evaluation on a real Twitter dataset, which shows the effectiveness of our method. The remainder of this paper is organized as follows. Section II surveys related work, whereas Section III presents the filtering rules, the inference problem statement and the evolutionary approach. Section IV discusses the results of the experimental evaluation on a real Twitter dataset. Finally, Section V concludes the paper.

II. Related work

Several approaches for reducing information overload in Twitter have been so far presented. In general, all of them propose recommendation systems that, according to different strategies (e.g., collaboration [8], semantic techniques [9], probabilistic models [10]), aim to recommend a selection of tweets that better fits the user's interests. All these solutions are interesting, but do not consider users' preferences. Rather than a system dependent solution, where the filtering strategy is decided by the service manager, we are interested to help the user to avoid tweets that, according to his/her sensibility and preferences, he/she might feel not relevant or inappropriate. In this sense, the issues to be investigated w.r.t. those in recommendation systems are different as these require to: (i) identify a proper language for expressing filtering rules in Twitter and (ii) design a tool able to infer filtering rules.

Additionally, tweets filtering has been investigated also to cope with the problem of Twitter spam. Here, several methods have been presented to detect spammers in Twitter (e.g., [11], [12], [13], [14] just to mention some of them). However, these

[1]The term has been popularized by Alvin Toffler in his book Future Shock, 1970.

proposals are mainly based on the characteristics of social networks and do not consider users preferences on filtering rules. To the best of our knowledge, a language for filtering posts in social network has been proposed only for Facebook in [15]. The proposed language makes users able to express constraints on post contents, on the profile of the author of the post, and on the relationship between the user specifying the rule and the post author. With respect to this work, which inspired our proposal of the filtering language, in this paper we also address the problem of filtering rules suggestion, acknowledging the difficulties the average users might have in modeling their filtering needs.

III. Our approach

A. Language for filtering policies

We define the language for specifying filtering policies basing on a simple model of Twitter posts. Let $\mathcal{T}$ be a statically defined set of *topics*, i.e., possible subjects of discussion being mentioned in a post. For instance, $\mathcal{T}$ may include "religion", "politics", and so on: the actual set of topics we considered in our experimentation is detailed in Section IV. Let $\mathcal{L}_P$ be a statically defined set of *post labels*, i.e., possible structural properties of a post expressed as binary attributes. We set $\mathcal{L}_P = \{\text{hasMedia}, \text{hasHashtags}, \text{hasURLs}\}$, which correspond, respectively, to a post containing an image or video, one or more hashtags, and one or more URLs. Let, finally, $\mathcal{L}_A$ be a statically defined set of *author labels*, i.e., possible binary attributes of the author of a post. We set $\mathcal{L}_A = \{\text{isVIP}\}$, which we set for Twitter users with $50\,000$ followers or more.

We associate each *Twitter post* p with a tuple $\langle T_P^p, T_A^p, L_P^p, L_A^p \rangle$ where: $T_P^p \subseteq \mathcal{T}$ is the set containing all the topics mentioned in p; $T_A^p \subseteq \mathcal{T}$ is the set containing all the *usual topics* of the author of the post (see below); $L_P^p \subseteq \mathcal{L}_P$ is the set of the labels of p; and, finally, $L_A^p \subseteq \mathcal{L}_A$ is the set of the labels of the author of the post—all 4 sets may be empty. Without loss of generality, we assume that the *usual topics* of an author are those topics which are mentioned in at least $\frac{1}{3}$ of the posts of that author.

We assume that a user may specify the *filtering policy* for his/her timeline. The policy describes the posts that cannot appear on the timeline as a set of *filtering rules*, as follows.

We define a *filtering rule* r as a tuple $\langle o_{T_P}, T_P^r, o_{T_A}, T_A^r, o_{L_P}, L_P^r, o_{L_A}, L_A^r \rangle$, where $T_P^r \subseteq \mathcal{T}$, $T_A^r \subseteq \mathcal{T}$, $L_P^r \subseteq \mathcal{L}_P$, and $L_P^r \subseteq \mathcal{L}_P$ are (possibly empty) sets while $o_{T_P}, o_{T_A}, o_{L_P}, o_{L_A} \in \{\subseteq, \not\subseteq\}$ are set operators. A post p is *filtered* by a rule r if and only if all the following conditions are met: $T_P^r o_{T_P} T_P^p$, $T_A^r o_{T_A} T_A^p$, $L_P^r o_{L_P} L_P^p$, and $L_A^r o_{L_A} L_A^p$. A *filtering policy* ρ is a set of rules. A post p is filtered by a policy if p is filtered by at least one rule of the policy.

As an example, a policy which filters all vulgar posts, all the posts concerning politics not authored by users who usually tweet about politics, and all the posts concerning sex containing some media and not authored by a VIP user would be defined as $\rho = \{r_1, r_2, r_3\}$ where:

$$r_1 = \langle \subseteq \{\text{vulgarity}\}, \subseteq \emptyset, \subseteq \emptyset, \subseteq \emptyset \rangle$$
$$r_2 = \langle \subseteq \{\text{politics}\}, \not\subseteq \{\text{politics}\}, \subseteq \emptyset, \subseteq \emptyset \rangle$$
$$r_3 = \langle \subseteq \{\text{sex}\}, \subseteq \emptyset, \subseteq \{\text{hasMedia}\}, \not\subseteq \{\text{isVIP}\} \rangle$$

For ease of reading, we write filtering rules by omitting the comma between the set operator and the corresponding set. Note that $A \subseteq \emptyset$ is satisfied for any set A.

B. Policy inference: problem statement

The language introduced in the previous section allows defining rather complex filtering policies. On the other hand, the high expressiveness of the language would require users to have both knowledge of the language and the ability to model their filtering needs in a way which can be described with the language. A tool capable of synthesizing filtering policies automatically, starting from (possibly few) examples of the desired filtering behavior would be highly useful.

We specify the *policy inference problem* as follows. Given a non-empty set P_+ of posts which should be filtered and a non-empty set P_- of posts which should not be filtered, a policy $\rho^\star$ has to be generated such that: (a) all posts of P_+ are filtered by $\rho^\star$, (b) no post of P_- is filtered by $\rho^\star$, and (c) $\rho^\star$ exhibits the minimal complexity. We quantify the complexity of a policy with the number $|\rho|$ of rules in the policy but more elaborate complexity measures could be used, e.g., the sum of the sizes of sets composing each rule.

We assess the quality of an inferred policy ρ on a pair $P_+^{\text{test}} \supseteq P_+, P_-^{\text{test}} \supseteq P_-$ of larger sets by means of *False Acceptance Rate (FAR)*, i.e., the ratio between the number of posts of P_+^{test} which are not filtered by ρ and the size of P_+^{test}; and *False Rejection Rate (FRR)*, i.e., the ratio between the number of posts of P_-^{test} which are filtered by ρ and the size of P_-^{test}.

C. Policy inference: evolutionary approach

We propose an approach based on a form of *separate-and-conquer* strategy [16] and inspired by [7]. The approach constructs a policy iteratively, one rule at a time. At each iteration an evolutionary search constructs a rule with minimal FAR and FRR on P_+ and P_-. Posts of P_+ which are already filtered by rules obtained so far are removed before executing the next iteration.

1) Evolutionary search: The evolutionary search takes a pair P_+, P_- of post sets as input and outputs a single rule $r^\star$. A population of candidate rules (*individuals*) is iteratively evolved by combining individuals with *genetic operators* until a termination condition is met. Upon the last iteration, the best individual $r^\star$ is chosen as outcome of the search.

We designed genetic operators tailored to our application domain, as follows. We designed a category of *mutation* operators (r denotes the individual to be mutated while r' denotes the mutated individual): (i) *set operator flip,* r' is a copy of r in which one among the 4 set operators chosen at random is flipped; (ii) *item addition,* r' is a copy of r in which,

in one among the 4 sets chosen at random, an item randomly chosen from the corresponding domain is inserted; (iii) *item removal,* r' is a copy of r in which, in one among the 4 sets chosen at random, an item randomly chosen is removed. We designed a category of *crossover* operators (r_1 and r_2 denote the individuals to which the operator is applied and r' denotes the resulting individual): (i) *item donation,* r' is a copy of r_1 in which, in one among the 4 sets chosen at random, a randomly-chosen item is inserted from the corresponding set of r_2; (ii) *set operator donation,* r' is a copy of r_1 in which one among the 4 set operators chosen at random is set to the corresponding set operator of r_2.

Each individual r is associated with a counter c_r, initially set to 1 and with a *fitness* $f(r) = \langle \text{FRR}(r), \text{FAR}(r), |r| \rangle$, where $\text{FRR}(r)$ is the FRR of r on P_-, $\text{FAR}(r)$ is the FAR of r on P_+, and $|r|$ is defined as the sum of the sizes of the sets defining r, i.e., $|r| = |T_P^r| + |T_A^r| + |L_P^r| + |L_A^r|$. The fitness of an individual measures the ability of the rule represented by the individual to satisfy the requirements of the policy inference problem described in the previous section.

Whenever two or more individuals are to be ranked based on their fitness, we use a lexicographic criterion: the individual with the lowest FRR comes first (i.e., is better); in case of tie, the one with the lowest FAR comes first; in case of tie, the one with smaller $|r|$ comes first—in multi-objective evolutionary optimization this ranking method is known as multi-layered fitness [17]. The rationale for this design choice is twofold: first, it is consistent with the policy composition procedure (see next section); second, we consider the case in which a "good" post is filtered worse than the case in which a "bad" post is not filtered.

Initially, the population is composed of one rule for each element in P_+: for each post $p = \langle T_P^p, T_A^p, L_P^p, L_A^p \rangle \in P_+$ an individual $r = \langle \subseteq T_P^p, \subseteq T_A^p, \subseteq L_P^p, \subseteq_A^p \rangle$ is constructed and inserted into the population. The population is then evolved through the following iterative procedure:

1) choose randomly whether to apply a mutation operator or a crossover operator, respectively with probability p_{mutation} and $1 - p_{\text{mutation}}$;
2) choose the specific operator among those in the selected category, with uniform probability;
3) choose one (with mutation) or two (with crossover) individuals from the population, using a *tournament selection*, i.e., by selecting $n_{\text{tournament}}$ individuals in the population at random and then picking the one with best fitness;
4) apply the operator to the chosen individual(s) to obtain a new individual r': if r' is present in the population, than increment its counter $c_{r'}$ by 1, otherwise, evaluate its fitness $f(r')$ and insert it in the population with $c_{r'} := 1$;
5) if the population size exceeds n_{pop}, then repeatedly remove the individual with worst fitness until the population size is less than or equal to n_{pop}.

The procedure is iterated until one of the following conditions is satisfied: (a) the number of fitness evaluation exceeds n_{eval} or (b) the counter $c_{r^\star}$ of the best individual exceeds n_{stop}. The rule in the final population with best fitness is then selected as output of the evolutionary search. Note that p_{mutation}, $n_{\text{tournament}}$, n_{pop}, n_{eval}, and n_{stop} are parameters of the search.

2) Policy composition: The policy composition procedure takes a pair P_+, P_- of post sets as input and outputs a full filtering policy ρ. The procedure consists of a sequence of evolutionary searches each one operating on a different input, as follows.

Initially, we set $\rho = \emptyset$ and $P'_+ = P_+$. Then, we execute the following steps:

1) execute an evolutionary search on P'_+, P_- and obtain a rule $r^\star$;
2) if $\text{FRR}(r^\star) = 0$ on P_- and $\text{FAR}(r^\star) < 1$ on P'_+, then add $r^\star$ to ρ, otherwise, terminate the procedure;
3) set P'_+ to the set of posts of P_+ which are *not* filtered by the current policy ρ;
4) if $P'_+ \neq \emptyset$, then restart from step 1, otherwise, terminate the procedure.

The rationale of the procedure is attempting to generate, at each iteration, one rule $r^\star$ that filters at least one new post of P_+ which is not filtered by the current policy (second condition of step 2). Since $r^\star$ does not filter any post in P_- (first condition of step 2) and rules are or-ed when applying a filtering policy, it follows that the resulting policy ρ will have: (a) $\text{FRR}(\rho) = 0$, and (b) FAR which is lower or equal than the one of its composing rules, i.e., $\forall r \in \rho, \text{FAR}(\rho) \leq \text{FAR}(r)$.

IV. Experimental evaluation

We aimed at both (a) verifying the ability of our proposed language to express filtering policies of realistic complexity and (b) experimentally assessing the effectiveness of our inference method in generating a policy from examples of the desired filtering behavior.

To this end, we collected a dataset of Twitter posts which we processed in order to annotate each post as required in our model, i.e., with topics and attributes. We first assembled a large set of 2 156 344 Twitter posts authored by 11 254 different Twitter users, by means of an automatic procedure exploiting the Twitter APIs. Then, we discarded non-English posts and randomly selected a subset of the most active users, obtaining a set composed of 27 089 posts authored by 3877 users. We applied the EgoCentric text classifier [18] to each of the 27 089 posts in order to associate each post with its topics: EgoCentric was specifically designed to operate on short text and pre-configured to assign topics in the set $\mathcal{T}$ including vulgarity, religion, politics, sex, work, alcohol, school, holiday, and health. Finally, we selected a subset P composed of 1707 posts including all the 707 posts with at least one topic and 1000 randomly-selected posts with no topic.

Having constructed the dataset P, we defined 5 different filtering scenarios. For each scenario, we manually built a *target policy* $\rho^\star$ representing specific filtering needs. The scenarios differ in complexity of the corresponding target policies, which range from $|\rho^\star| = 1$ to $|\rho^\star| = 4$. We applied each $\rho^\star$ to P obtaining sets P_+^0 and P_-^0 that contain, respectively, posts which are filtered and which are not filtered by $\rho^\star$.

#	$\lvert\rho^\star\rvert$	$\lvert P_+^0\rvert$	$\lvert P_-^0\rvert$	On P_+, P_- FRR	On P_+, P_- FAR	On $P_+^{\text{test}}, P_-^{\text{test}}$ FRR	On $P_+^{\text{test}}, P_-^{\text{test}}$ FAR	$\lvert\rho\rvert$
1	1	110	1597	0.00	0.00	0.00	0.00	1
2	1	9	1698	0.00	0.00	0.00	0.00	1
3	2	196	1511	0.00	0.00	0.00	0.00	3
4	3	166	1541	0.00	0.00	0.00	0.00	3
5	4	32	1675	0.00	0.00	0.00	0.06	2
Avg.				0.00	0.00	0.00	0.01	

TABLE I: Results for $l = 0.5$. Each row corresponds to a different filtering scenario.

We executed 3 experiments for each of the 5 scenarios by varying the proportion of posts of P_+^0 and P_-^0 used as examples. We set $l \in \{1, 0.5, 0.1\}$, where $l = \frac{|P_+|}{|P_+^0|} = \frac{|P_-|}{|P_-^0|}$. Each experiment consisted of an execution of our inference method on sets of examples P_+, P_- constructed by random sampling followed by measuring FAR and FRR of the resulting policy on $P_+^{\text{test}} = P_+^0$, $P_-^{\text{test}} = P_-^0$. We repeated each experiment 3 times, with different random seeds, and averaged the results of each experiment across the 3 repetitions. We executed our inference method with $p_{\text{mutation}} = 0.5$, $n_{\text{tournament}} = 3$, $n_{\text{pop}} = 100$, $n_{\text{eval}} = 2500$, and $n_{\text{stop}} = 100$, as suggested in [7]. Execution of each experiment took a few seconds on commodity hardware.

Table I presents the results for $l = 0.5$. It can be seen that the policy inferred by our method automatically is indeed able to capture the examples (FAR and FRR are always equal to 0 on sets P_+, P_-). Most importantly, results on testing data demonstrate that the policy is able to *generalize* beyond the examples available for inference: FAR in scenario 5 is very small with all the other indexes corresponding to an ideal filtering behavior. Data for $l = 1, 0.1$ allow drawing the very same conclusions: FAR = FRR = 0 on the examples; FRR = 0 on $P_-^{\text{test}} = P_-^0$; average FAR on P_+^{test} is 0 and 0.08, respectively.

V. Concluding remarks and future work

We have investigated the feasibility of an approach for enabling users to define personalized strategies for filtering Twitter posts, i.e., for defining which kinds of tweets should not appear on their timeline according to their interests, preferences and sensibility. This problem is becoming increasingly relevant due to the huge amount of information that is shared and disseminated on online social networking services.

We have proposed a language that, despite its simplicity, allows defining rather complex filtering policies and a tool capable of synthesizing filtering policies automatically, based solely on examples of the desired filtering behavior. The experimental evaluation on a real Twitter dataset and 5 different filtering scenarios has suggested that the proposed approach is indeed able to generate filtering policies that are effective and that generalize beyond the provided examples. Furthermore, the time for inferring a policy is so short to allow devising an implementation of the approach in the form of an interactive tool, e.g., as a browser plugin.

Although our preliminary assessment is certainly to be validated on a larger scale, we believe that our results are highly promising.

References

[1] M. Gomez Rodriguez, K. Gummadi, and B. Schölkopf, "Quantifying information overload in social media and its impact on social contagions," in *Proceedings of the Eighth International Conference on Weblogs and Social Media*. AAAI Press, 2014, pp. 170–179.

[2] K. Bontcheva, G. Gorrell, and B. Wessels, "Social media and information overload: Survey results," *CoRR*, vol. abs/1306.0813, 2013.

[3] G. C. Akcora and E. Ferrari, *Encyclopedia of Social Network Analysis and Mining*. New York, NY: Springer New York, 2014, ch. Graphical User Interfaces for Privacy Settings, pp. 648–660. [Online]. Available: http://dx.doi.org/10.1007/978-1-4614-6170-8_360

[4] N. Hu, P. G. Bradford, and J. Liu, "Applying role based access control and genetic algorithms to insider threat detection," in *Proceedings of the 44th annual Southeast regional conference*. ACM, 2006, pp. 790–791.

[5] Y. T. Lim, P. C. Cheng, P. Rohatgi, and J. A. Clark, "MLS security policy evolution with genetic programming," in *Proceedings of the 10th annual conference on Genetic and evolutionary computation*. ACM, 2008, pp. 1571–1578.

[6] Y. T. Lim, P.-C. Cheng, P. Rohatgi, and J. A. Clark, "Dynamic security policy learning," in *Proceedings of the first ACM workshop on Information security governance*. ACM, 2009, pp. 39–48.

[7] E. Medvet, A. Bartoli, B. Carminati, and E. Ferrari, "Evolutionary inference of attribute-based access control policies," in *Evolutionary Multi-Criterion Optimization*. Springer, 2015, pp. 351–365.

[8] J. Hannon, M. Bennett, and B. Smyth, "Recommending twitter users to follow using content and collaborative filtering approaches," in *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010, pp. 199–206.

[9] P. Kapanipathi, F. Orlandi, A. P. Sheth, and A. Passant, "Personalized filtering of the twitter stream," in *Proceedings of the 10th International Semantic Web Conference, Bonn, Germany, October 23-27*, 2011.

[10] Y. Kim and K. Shim, "Twitobi: A recommendation system for twitter using probabilistic modeling," in *Proceeding of the IEEE 11th International Conference on Data Mining (ICDM)*. IEEE, 2011, pp. 340–349.

[11] X. Zhang, Z. Li, S. Zhu, and W. Liang, "Detecting spam and promoting campaigns in twitter," *ACM Trans. Web*, vol. 10, no. 1, pp. 4:1–4:28, Feb. 2016. [Online]. Available: http://doi.acm.org/10.1145/2846102

[12] C. Freitas, F. Benevenuto, S. Ghosh, and A. Veloso, "Reverse engineering socialbot infiltration strategies in twitter," in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, ser. ASONAM '15. New York, NY, USA: ACM, 2015, pp. 25–32. [Online]. Available: http://doi.acm.org/10.1145/2808797.2809292

[13] S. Sedhai and A. Sun, "Hspam14: A collection of 14 million tweets for hashtag-oriented spam research," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '15. New York, NY, USA: ACM, 2015, pp. 223–232. [Online]. Available: http://doi.acm.org/10.1145/2766462.2767701

[14] C. Yang, R. Harkreader, J. Zhang, S. Shin, and G. Gu, "Analyzing spammers' social networks for fun and profit: A case study of cyber criminal ecosystem on twitter," in *Proceedings of the 21st International Conference on World Wide Web*, ser. WWW '12. New York, NY, USA: ACM, 2012, pp. 71–80. [Online]. Available: http://doi.acm.org/10.1145/2187836.2187847

[15] M. Vanetti, E. Binaghi, E. Ferrari, B. Carminati, and M. Carullo, "A system to filter unwanted messages from osn user walls," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 2, pp. 285–297, Feb 2013.

[16] J. Fürnkranz, "Separate-and-conquer rule learning," *Artificial Intelligence Review*, vol. 13, no. 1, pp. 3–54, 1999.

[17] J. Eggermont, J. N. Kok, and W. A. Kosters, "Genetic programming for data classification: Partitioning the search space," in *Proceedings of the 2004 ACM symposium on Applied computing*. ACM, 2004, pp. 1001–1005.

[18] W. Lucia and E. Ferrari, "Egocentric: ego networks for knowledge-based short text classification," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 2014, pp. 1079–1088.