Identifying Retweetable Tweets with a Personalized Global Classifier

Michail Vougioukas
National Center of Scientific Research "Demokritos"
Athens, Greece

Ion Androutsopoulos
Athens University of Economics and Business
Athens, Greece

Georgios Paliouras
National Center of Scientific Research "Demokritos"
Athens, Greece

ABSTRACT

In this paper we present a method to identify tweets that a user may find interesting enough to retweet. The method is based on a global, but personalized classifier, which is trained on data from several users, represented in terms of user-specific features. Thus, the method is trained on a sufficient volume of data, while also being able to make personalized decisions, i.e., the same post received by two different users may lead to different classification decisions. Experimenting with a collection of approx. 130K tweets received by 122 journalists, we train a logistic regression classifier, using a wide variety of features: the content of each tweet, its novelty, its text similarity to tweets previously posted or retweeted by the recipient or sender of the tweet, the network influence of the author and sender, and their past interactions. Our system obtains $F_1 \approx 0.9$ using only 10 features and 5K training instances.

KEYWORDS

Social networks, social media, Twitter, personalization, user modeling, filtering, recommendation, machine learning, evaluation.

ACM Reference format:
Michail Vougioukas, Ion Androutsopoulos, and Georgios Paliouras. 2018. Identifying Retweetable Tweets with a Personalized Global Classifier. In *Proceedings of 10th Hellenic Conference on Artificial Intelligence, Rio Patras, Greece, July 9–15, 2018 (SETN '18),* 8 pages.
DOI: 10.1145/3200947.3201019

1 INTRODUCTION

Information shared on social networks is ever increasing and users are often overwhelmed by the number of posts (e.g., tweets) they receive. Many of the incoming posts are of marginal or no interest to their recipients. Consequently, interesting posts may be ignored or overlooked by time-constrained users, who may also give up reading their timelines. Filters that estimate the interest of each incoming post can alleviate this problem, for example by allowing users to sort incoming posts by predicted interest (e.g., 'top stories' vs. 'most recent' in Facebook) or by mixing recent posts with predicted interesting ones (e.g., 'in case you missed it' in Twitter).

SETN '18, Rio Patras, Greece

978-1-4503-6433-1/18/07…$15.00
DOI: 10.1145/3200947.3201019

There have been two main approaches to detect interesting posts in social networks: *global* filters [1, 2, 19] and *personal* filters [6, 15, 18]. Global filters try to predict how interesting a post is for the entire social network or at least a broad audience. A single global filter is typically trained on a large collection of posts and the reactions of all users to each post (e.g., total number of retweets per post). The trained global filter is then used to assign a single, user-independent interest score to each new post. By contrast, personal filters are typically trained on posts received by a particular user and the reactions of the particular user (e.g., whether or not the user retweeted each post). A separate filter is trained per user and is then employed to provide user-specific interest scores for each tweet or, generally, social post. Personal filters can, at least in principle, provide recommendations tailored to a particular user's own interests, which may not coincide with the interests of the majority of users that global filters are trained to predict. On the other hand, global filters are typically trained on much larger datasets compared to personal filters. Hence, global filters may work better in practice, especially with new users, for which personal filters may have very few training instances (the 'cold start' problem).

Following Uysal and Croft [14] and Zhang et al. [21], in this paper we investigate a hybrid approach that attempts to combine the strengths of both global and personal filters. As in global filters, we train a *single* system on a large collection of tweets received by multiple users. Each tweet, however, is represented as a feature vector that includes *user-specific features* (Fig. 1), for example indicating the extent to which the incoming tweet is similar to tweets previously posted or retweeted by the recipient, or how often the recipient has retweeted posts of the sender of the tweet. If the same tweet is received by two different users, it will be represented by two different feature vectors. This allows the system to take into account user preferences and produce different predictions per recipient, even for the same incoming tweet, as in personal filters, while still being able to generalize over different users (e.g., learn that users are in general more likely to retweet posts that are similar to their own posts). We train a single shared logistic regression model for all users, in order to predict if a tweet received by a particular user will be retweeted by that user or not. We examine the effect of several types of features that examine the content of each incoming tweet, the similarity of the incoming tweet to tweets previously posted or retweeted by the recipient or the sender, the network influence of the sender and recipient, the interaction between them (e.g., if they have mentioned each other in previous tweets), the novelty of the incoming tweet (e.g., its similarity to tweets recently seen by the recipient). On a dataset of approx. 130K tweets received by 122 journalists, our system obtains $F_1 \approx 0.9$ using only 10 features and approximately 5K training instances.

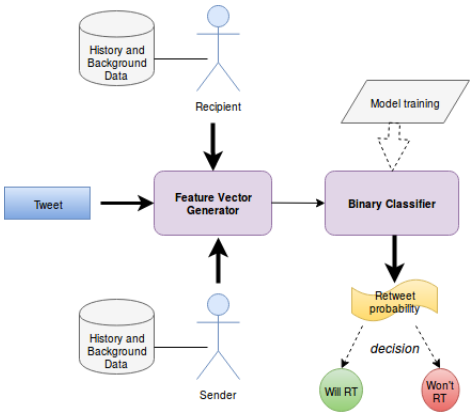

Figure 1: Architecture of our system.

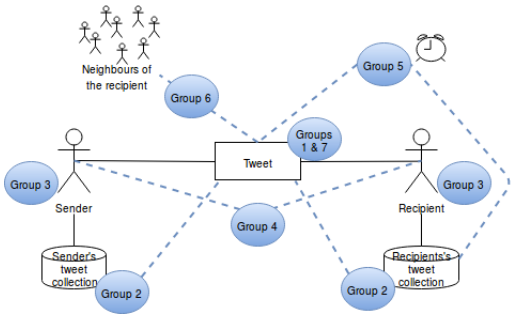

Figure 2: Groups of features used by our system and how they relate to the tweet itself, the sender, the recipient etc.

Using previous retweet (and non-retweet) actions as gold labels has the advantage that no extra human labeling is required to construct training and test data, as opposed to asking users to label their incoming tweets with interest scores. On the other hand, retweeting is only an approximate signal of interest, as users do not retweet all the posts they find interesting. Nevertheless, retweeting is usually an indication of great interest in a post and, hence, our system can be used to detect tweets that a particular user would find very interesting (interesting enough to retweet), which could then be ranked higher or mixed with recent tweets.

The main contributions of this paper are: (a) a lightweight prediction model, which attains high F1 score with a small number of features and training instances; (b) investigation of most candidate features mentioned in related literature and variants thereof, grouped into feature types for further research; (c) a large dataset of tweets and associated user information, which we have made publicly available in an encoded form.[1]

Section 2 below describes our system. Section 3 presents the experiments we performed. Section 4 discusses related work. Section 5 concludes and proposes future work.

2 SYSTEM DESCRIPTION

2.1 System overview

Our system predicts how likely it is that a particular user (the *recipient* of Fig. 1) will retweet a particular incoming tweet. The system also has access to the history of the recipient (e.g., tweets the recipient has previously received or posted), the history of the sender of the tweet, as well as background information about the recipient and the sender (e.g., number of followers).[2] By *sender* we mean the user that caused the recipient to receive the tweet, either by authoring it directly (if the recipient follows the author) or by retweeting it (if the recipient does not follow the author). The tweet is represented as a feature vector, which includes features that depend on the particular recipient; hence, the same tweet will be represented by a different feature vector when the system tries to estimate if another recipient will retweet it or not. The feature vector is passed on to a (binary) logistic regression classifier that predicts if the recipient will retweet the incoming tweet or not. The classifier (one model for all recipients) is trained on tweets received by Twitter users and the users' reactions (whether they retweeted the incoming tweets or not).[3]

2.2 Preprocessing of the tweet text

Before further processing, the text of each tweet is normalized as follows to allow the classifier to generalize (e.g., over different URLs, different numbers, smileys that express the same sentiment).

(1) All URLs are replaced by the same pseudo-token (e.g., '_url_'), which denotes a generic URL.
(2) All numbers are replaced by a pseudo-token (e.g., '_num_').
(3) Each type of smiley is replaced by a different pseudo-token:
  (a) Love/like smileys (e.g., '<3').
  (b) Positive sentiment smileys (e.g., ':-)').
  (c) Negative sentiment smileys (e.g., ':-(').
  (d) Neutral sentiment smileys (e.g., ':-|').
(4) All tokens are converted to lower case.

These steps are based on the preprocessing used in GloVe [12] to turn words into embeddings [11]. Hence, in a future extension of our system one could easily use GloVe embeddings.

2.3 Features used by the classifier

The feature vector of each incoming tweet contains up to 50 features, each corresponding to a factor that we suspect may help predict if the tweet will be retweeted or not. The features were constructed by taking into account previous related work (Section 4), the information provided by Twitter's API, and our own experience as Twitter users. The 50 features are divided into 7 groups.

Group 1 (Fig. 2, Table 1) contains features that examine the tweet itself (e.g., length, if it contains a URL or not, if it mentions a Twitter account). Longer tweets, or tweets that contain URLs of longer posts (e.g., news articles) or photographs may be more informative and, thus, more interesting. Tweets that mention other user accounts may be parts of dialogues, which may be uninteresting to recipients, unless they interact frequently with the sender (see also Group 4). Hashtags may indicate trending topics. Tweets that have already

[1] Dataset available at: http://nlp.cs.aueb.gr/publications.
[2] We use Twitter's API (https://dev.twitter.com/rest/public) to obtain this information.
[3] We used Weka's implementation of logistic regression (http://www.cs.waikato.ac.nz/ml/weka/), with default hyper-parameter values. Modifying the defaults had no significant effect in preliminary experiments.

been retweeted or favoured by many users are more likely to be important. Exclamation marks indicate surprise or strong feelings.

Group 2 (Fig. 2, Table 2) contains features that examine how similar the incoming tweet is to particular collections of tweets (e.g., all tweets previously posted by the sender). The similarity between the incoming tweet t and a collection of tweets $C = \{c_1, \ldots, c_n\}$ is computed as the average TF-IDF cosine similarity between t and each c_i. The intuition in Group 2 is that recipients may prefer tweets that are similar or dissimilar (if they prefer surprising posts) to the posts of the particular sender, or their own posts, or the posts they usually see or retweet.

Group 3 (Fig. 2, Table 3) contains features modeling the network influence, popularity, and authority of the sender and the recipient. These features include Twitter account statistics (number of followers, number of posts, days active for, list subscriptions), features that may indicate authority (verified accounts, URLs in the description fields of their profiles), as well as scores obtained from Klout, a service that estimates a user's social influence by taking into account their activity in various social networks.[4]

Group 4 (Fig. 2, Table 4) contains features that capture the interaction between the sender and the recipient (e.g., whether or not tweets of the sender mention the recipient). The intuition is that recipients are more likely to be interested in posts of senders they interact more closely with.

Group 5 (Fig. 2, Table 5) contains features that attempt to estimate the timeliness of the incoming tweet. A tweet that is very similar to other recently received or retweeted tweets may be old news. The similarity scores of these features are again averaged TF-IDF cosine similarities.

Group 6 (Fig. 2, Table 6) contains features related to the users the recipient follows (the user's *neighbours*). The neighbours presumably have common interests with the recipient. Hence, if the original author of the incoming tweet is a neighbour of the recipient or if the incoming tweet has been retweeted by many neighbours of the recipient, this may be an indication that the recipient will also find the incoming tweet interesting.

Group 7 (Fig. 2, Table 7) complements the features of Group 1 by looking for particular keywords and parts of speech (nouns, verbs, articles) in the incoming tweet.[5] The features of Group 7 are based on the work of Tan et al. [13], who found that the wording of a post significantly affects its propagation, compared to other posts that express the same information using different wordings. Tan et al. provide a list of 20 'good' keywords, believed to increase the propagation probability of a post, and 20 'bad' keywords.

3 EXPERIMENTS

3.1 Dataset

In our experiments, the recipients (Fig. 1 and 2) were 122 journalists. We started with a list of 262 journalists, available from previous work [20], but we retained only journalists that write in English.[6] We also discarded journalists for which we could not collect at least 500 retweets, ending up with 122 journalists. The dataset of our experiments consists of 122 subsets, one for each

Table 1: Features of Group 1 (the tweet itself).

Feature ID	Feature Description
FT1	Tweet length in characters.
FT2	Does the tweet contain a URL?
FT3	Does it mention a Twitter account (@username)?
FT4	Does it contain a hashtag?
FT5	Global retweet count (times it has been retweeted).
FT6	Global favourite count.
FT7	Does the tweet contain an exclamation mark?
FT8	Does it contain a photo?
FT9	Number of Twitter accounts it mentions.

Table 2: Features of Group 2 (average TF-IDF cosine similarity of the tweet to other tweet collections).

Feature ID	Feature Description
FT10	Similarity to tweets previously posted (authored or retweeted) by the sender.
FT11	Similarity to tweets previously posted (authored or retweeted) by the recipient.
FT12	Similarity to tweets previously seen by the recipient (excluding 'easy' negative tweets and tweets from recently inactive neighbours – see Section 3.1).
FT13	Similarity to previous retweets of the recipient.

Table 3: Features of Group 3 (influence, popularity, authority of the sender and recipient).

Feature ID	Feature Description
FT14	Number of users that follow the sender.
FT15	Number of users the sender follows.
FT16	Number of tweets the sender has posted (authored or retweeted).
FT17	Number of curated lists the sender subscribes to.
FT18	Is the sender a verified account?
FT19	Days the sender's account has been active for.
FT20	Does the sender have a URL in their description?
FT21	The Klout score (influence) of the sender.
FT22	Delta of FT21 from the previous 24 hours.
FT23	Delta of FT21 from the previous 7 days.
FT24	Delta of FT21 from the previous 30 days.
FT25	Number of users that follow the recipient.
FT26	Number of users the recipient follows.
FT27	Number of tweets the recipient has posted.
FT28	Number of curated lists the recipient subscribes to.
FT29	Is the recipient a verified account?
FT30	Days the recipient's account has been active for.
FT31	Does the recipient have a URL in their description?
FT32	The Klout score of the recipient.
FT33	Delta of FT32 from the previous 24 hours.
FT34	Delta of FT32 from the previous 7 days.
FT35	Delta of FT32 from the previous 30 days.

[4] See http://klout.com/. All the features are normalized to [0, 1].
[5] We use CMU ARK Twitter tagger [8] (http://www.cs.cmu.edu/~ark/TweetNLP/).
[6] We used a flag in Twitter's API to detect the language.

Table 4: Features of Group 4 (sender-recipient interaction).

Feature ID	Feature Description
FT36	Is the recipient mentioned (@username) in the incoming tweet?
FT37	Has the sender ever mentioned the recipient?
FT38	Has the recipient ever mentioned the sender?
FT39	Has the sender ever retweeted the recipient?
FT40	Has the recipient ever retweeted the sender?
FT41	No. of times the recipient has retweeted the sender.

Table 5: Features of Group 5 (timeliness of incoming tweet).

Feature ID	Feature Description
FT42	Similarity to tweets seen by the recipient during the previous week (excluding 'easy' negative tweets and tweets from recently inactive neighbours).
FT43	Similarity to tweets retweeted by the recipient during the previous week.

Table 6: Features of Group 6 (neighbours of the recipient).

Feature ID	Feature description
FT44	Is the author of the incoming tweet a neighbour of the recipient? (The sender may be the author of the tweet or a neighbour that retweeted it. In the latter case, the original author may not be a neighbour.)
FT45	Number of times the incoming tweet has been retweeted by the neighbours of the recipient.

Table 7: Features of Group 7 (wording of the tweet).

Feature ID	Feature Description
FT46	Number of keywords in the incoming tweet explicitly asking to retweet/share (e.g., 'RT', 'spread', 'share').
FT47	Number of nouns and verbs in the incoming tweet.
FT48	Number of definite articles in the incoming tweet.
FT49	Number of indefinite articles in the incoming tweet.
FT50	Number of 'good' keywords minus number of 'bad' keywords in the tweet, using the keywords of [13].

journalist. Each subset comprises the most recent retweets of the corresponding journalist that we could collect through Twitter's API. The number of retweets in each subset was at least 500 and at most 2,500.[7] In each subset, the journalist's retweets are treated as *positive instances*.

Each subset also contains *negative instances*, meaning incoming tweets that the journalist did not retweet. To obtain the negative instances for each journalist we crawled the timelines of the users the journalist follows (neighbours) and collected their most recent posts (tweets authored or retweeted by the neighbour) that were not included in the positive instances of the journalist. To make the dataset more challenging, we excluded *'easy' negative instances*, meaning incoming tweets from neighbours that the journalist has never retweeted in the past, assuming that the journalist does not really care about posts from such neighbours. We also excluded negative instances from *recently inactive neighbours* (neighbours without any posts in the last seven days).

Our dataset was collected in late September 2015. To avoid using very old tweets, we discarded instances that were posted before January 2014. Hence, the dataset covers a period of approximately 19 months and contains approximately 12 million instances in total, involving 63,800 users (senders or recipients). Since the collected negative instances were many more than the positive ones, we randomly downsampled the negative instances of each journalist to obtain an equal number of positive and negative instances in each subset. This left a total of 133,000 instances (66,500 positive, 66,500 negative) in the 122 subsets.[8] To create training, development, and test sets, we first merged the 122 subsets and temporally ordered (by time posted) all the positive instances and, separately, all the negative instances. We removed all incoming duplicates per receiver (e.g., same tweet reaching the same receiver at different times via retweets of different senders the receiver follows), keeping only the earliest among duplicates.

We then formed 140 temporally ordered *batches*. Batch 1 contains the earliest 475 positive and the earliest 475 negative of the 133,000 instances. Batch 2 contains the next 475 positive and the next 475 negative instances etc.[9] The first 120 batches were used as the *training set* (57,000 positive and 57,000 negative instances), the next 10 batches were used as the *balanced development set* (4,750 positive and 4,750 negative instances), and the last 10 batches were used as the *balanced test set* (4,750 positive and 4,750 negative instances). We also constructed alternative, *unbalanced development and test sets* by randomly downsampling the positive (retweeted) instances in each batch of the balanced development and test sets, leaving 25 positive (5%) and 475 negative instances (95%) in each batch (250 positive and 4,750 negative instances in each unbalanced set).

We always train the logistic regression classifier of our system (Fig. 1) on the balanced training set. Using a balanced training set is common practice for discriminative supervised learning algorithms. Previous experiments [16] also indicated that training the logistic regression classifier on a balanced set leads to better performance on the development set, compared to using an unbalanced training set, even when the classifier is evaluated on an unbalanced development set with the same positive-to-negative ratio as the unbalanced training set. For a classifier trained on a balanced set, the balanced development and test sets are expected to be easier than their unbalanced counter-parts, since all the balanced sets have the same priors; this is also confirmed by our experimental results. The balanced development and test sets, however, are unrealistic, because they assume that receivers retweet on average half of their incoming tweets. The unbalanced development and test sets are intended to evaluate our system in a more realistic scenario, where receivers retweet only 5% of their incoming tweets.

To bypass privacy issues, the dataset which was used during our experiments was made publicly available in an encoded form, where words were replaced by unique integer identifiers, as in

[7] We could not collect more, due to restrictions of Twitter's API.

[8] IDF scores were estimated on the 12 million instances.

[9] The incoming tweets of the 122 journalists are distributed almost uniformly across the batches.

Table 8: Pearson correlation of the top 10 features to the class label (10-fold cross-validation on the training set).

Feature	Pearson	Feature Description
FT43	0.60	Similarity to tweets retweeted by the recipient during the previous week.
FT10	0.57	Similarity to tweets previously posted (authored or retweeted) by the sender.
FT21	0.49	The Klout score (influence) of the sender.
FT16	0.47	Number of tweets the sender has posted.
FT13	0.44	Similarity to tweets previously retweeted by the recipient.
FT45	0.42	Number of times the tweet has been retweeted by the recipient's neighbours.
FT44	0.40	Is the author of the incoming tweet a neighbour of the recipient?
FT40	0.40	Recipient ever retweeted the sender?
FT11	0.40	Similarity to tweets previously posted (authored or retweeted) by the recipient.
FT38	0.36	Recipient ever mentioned the sender?

previous spam filtering and legal text analytics datasets we have made available [3, 5].

3.2 Incremental training and evaluation

To study the effect of the size of the training set, each experiment was repeated 120 times, each time training the logistic regression classifier on the first (earliest) k batches of the training set ($1 \leq k \leq 120$), always using the same development or test set (10 batches each) to evaluate the performance of the classifier for each k value. We used *precision* (P), *recall* (R), and *F1 score* to evaluate the performance of the classifier, defined as usually.

3.3 Experiments on the development set

To get a first view of the usefulness of the features of Section 2.3, we ranked them by decreasing Pearson correlation [4] to the class label, using a 10-fold cross-validation on the training set (Section 3.1). The Pearson correlations of the top 10 features are shown in Table 8. Interestingly, the seven feature groups of Section 2.3 are not equally represented in the top 10 (Table 8). Only Group 2 (content similarity), Group 3 (influence, authority, popularity, but mostly of the sender), Group 4 (sender-recipient interaction), and Group 6 (neighbours) have features among the top 10.

We then evaluated the system with respect to its F1 score on the unbalanced development set, using an increasing number k of training batches ($1 \leq k \leq 120$), with different numbers of top-m features ($m \in \{1, 2, 10, 20, 35, 50\}$). The results of these experiments are shown in Fig. 3. A first observation is that the learning curves are steep for the first few training batches, but flatten out after approximately the first 12 batches (11,400 examples). This is a general trend for all of our experiments and suggests that a larger training set would not improve the system's performance.

A second observation is that the best results are obtained with the top 10 features (Fig. 3). Adding more features leads to increasingly worse results, possibly because the additional features add noise. Indeed, after the first 15-20 top features, the Pearson correlation of the features to the class label is quite low (<0.13). The performance

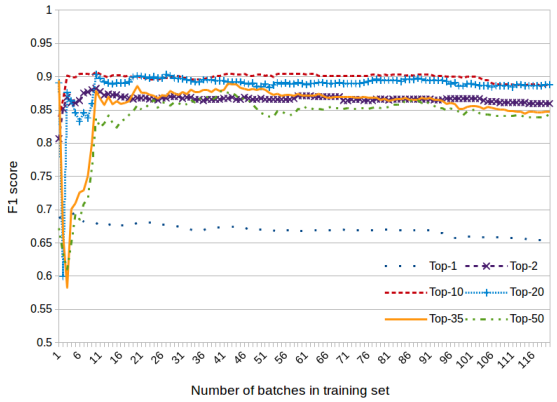

Figure 3: F1 on the unbalanced development set, for different numbers of top features.

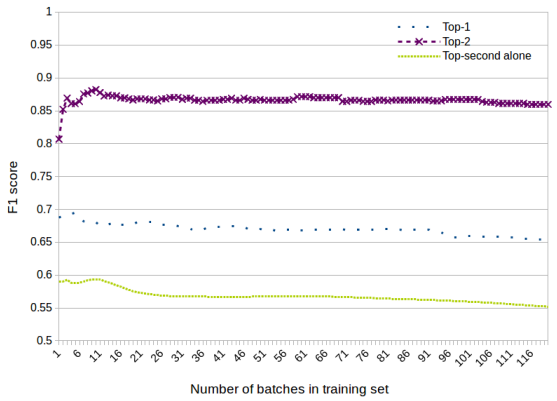

Figure 4: F1 on the unbalanced development set, using only the top feature (FT10), only the 2nd-top (FT43), or both.

of a 'lightweight' system with only the top two features (F1 ≈ 0.87) is comparable to that of the top 10 features (Fig. 3).

We investigated further the notable change in F1 when the second top feature is added to the top one (Fig. 3, curves Top-1 and Top-2). Figure 4 shows the F1 score, again on the unbalanced development set, using only the top feature (FT10), only the second-top (FT43), or both. The second-top feature alone is not a good predictor, but the combination of the two features increases F1.

Figure 5 sheds more light on the role of the top two features (FT10, FT43). It plots the positive and negative instances of a random subset (251 positive instances, 4,494 negatives) of the unbalanced development set. The straight line is the separator the logistic regression learned on the training set. In most cases, the line correctly separates the negative (stars) from the positive (crosses) instances, which agrees with the high F1 score in Figures 3 and 4.

As one might expect, most negative instances (stars) have low similarity (small values on the horizontal axis of Fig. 5) to the tweets the recipient retweeted during the previous week (FT43). This suggests that recent retweets of the recipients are good indicators of their current interests. Perhaps more unexpectedly, most positive

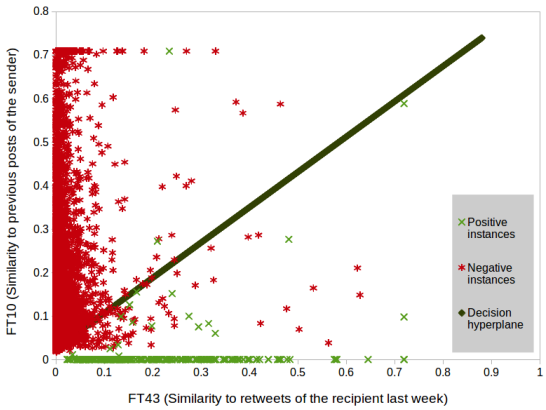

Figure 5: Sample positive and negative instances from the unbalanced development set and the linear separator the logistic regression classifier learned on the training set.

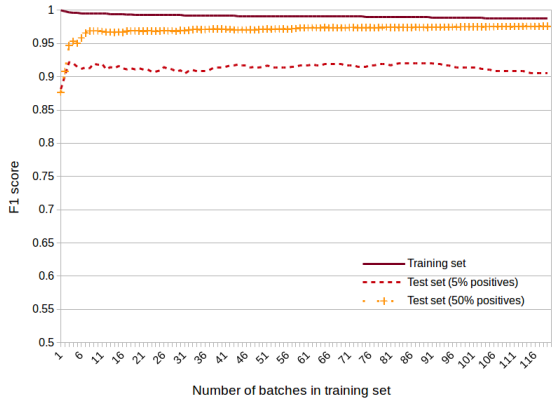

Figure 6: F1 on the balanced and unbalanced test set vs. F1 on the (always balanced) training set, using the top 10 features.

examples (crosses) have very *low* similarities to the previous posts of the sender (FT10). Intuitively, recipients tend to prefer (or at least retweet) posts that are *unusual* for the particular sender (posts that are surprisingly not about the usual topics of the sender, to the extent that TF-IDF cosine similarity captures topic similarity).

Figure 5 also illustrates the effect of combining the two features. Negative instances tend to have small values on the horizontal axis (FT43), but a non-negligible subset of positive instances also have small FT43 values. Most of those positive instances, however, have near-zero values on the vertical axis (FT10), unlike most negative instances and, hence, the combination of the two features improves classification accuracy. However, a non-linear classifier might manage to separate better the instances near the origin, where an S-shaped separator seems to be needed.

3.4 Experiments on the test set

In a final set of experiments, we evaluated our system on the (previously unseen) test set (10 fresh batches), using both the balanced (50% positives, 50% negatives) and the unbalanced (5% positives, 95% negatives) versions of the test set (Section 3.1). We used the top 10 features in these experiments, which had led to the best results on the development set (Section 3.3). The training set was the same as in the previous experiments (balanced). Fig. 6 shows the F1 scores on the two versions of the test set, along with the F1 scores on the batches of the training set the classifier has been trained on. The performance of a supervised classifier is typically better on the training data it has encountered, compared to its performance on unseen test data. Hence, the performance on the encountered training data is a boundary of the performance on test data. A large gap between the two is often due to overfitting the training data. The performance on the training data typically deteriorates as more training data are added, due to reduced overfitting.

Figure 6 shows the system performs better on the unbalanced test set (F1 $\approx$ 0.92) than on the unbalanced development set (cf. Fig 3). As expected, the system performs better on the balanced test set, which has the same positive-to-negative ratio (50% positives) as the training set, and worse on the unbalanced test set (5% positives). The gap between the performance on the training and balanced test data is small, indicating that the system does not significantly overfit the training data. The larger gap between the performance on the training and unbalanced test data is due to the change of ratio from the training to the test data, which makes the problem more difficult for the classifier. Again, both test curves flatten out after very few training batches (~5 for the unbalanced test set, ~12 for the balanced), though the balanced test F1 score continues to improve slowly, whereas the unbalanced test F1 does not.

4 RELATED WORK

4.1 Global filters for social media

Global filters aim to identify content which is interesting for a large audience. Yang et al. [19] used Latent Dirichlet Allocation (LDA) in a filter aiming to detect globally interesting tweets, as opposed to tweets that are only interesting to their direct recipients.

Hurlock and Wilson [10] investigated qualitative factors (e.g., reporting personal experience or not, providing specific information, timeliness, trusted author) that affect the perceived usefulness of the tweets returned by a search engine. Although they considered a different task (search) than the one we considered (predicting retweets) and their factors are not always easy to map to computable features (e.g., reporting personal experience, usefulness of a link), their work influenced our choice of features.

Duan et al. [7] used a learning-to-rank algorithm, experimenting with several types of features. They found that features related to the authority of senders (e.g., number of lists the author is included in) along with tweet length and presence of URL were particularly useful. These findings influenced our choice of features.

Alonso et al. [1, 2] considered several types of features and in their early work reported that a single feature (presence of URL) was enough to obtain 80% accuracy. Their later work [2], however, showed that human annotators did not agree on which tweets were interesting (inter-annotator agreement was as low as for random choices), concluding that interest is a subjective, not global notion.

4.2 Personal filters for social media

In previous work [15], we developed personal filters for Twitter, using the incoming tweets of six recipients, annotated with interest scores by the recipients themselves. Each filter was trained and tested on incoming tweets of a particular recipient, using the same learning algorithm and features. Manual annotation turned out to be a bottleneck and we could not obtain more than 1,000 annotated incoming tweets per recipient. Thus, we concluded that training a separate filter per user is not realistic and does not address the cold start problem, where a filter must be provided to a new user (recipient), with no training data available for this user.

Waldner and Vassileva [18] trained a different filter per Twitter user, using Naive Bayes. They classified incoming tweets in three classes (interesting, neutral, uninteresting) and studied user interface designs to emphasize 'interesting' tweets in timelines.

4.3 Hybrid personalized global filters

Uysal and Croft [14] consider two tasks: (a) predicting if an incoming tweet will be retweeted by a particular recipient or not and (b) ranking the potential recipients of a particular tweet so that recipients more likely to retweet it will be higher. We considered only the former task, but the same system could be used for the latter task too. The system of Uysal and Croft is hybrid, in the sense that it is global (a single filter for all users), but the feature vectors that represent the tweets include recipient-specific features, as in our own work. The features of Uysal and Croft are also similar to the ones we used. They consider the incoming tweet, the author, the recipient, their previous interaction etc. In fact, our feature set was largely based on that of Uysal and Croft, though we strived for engineering simplicity (e.g., we do not use personal language models), we included additional features (e.g., Klout scores, more similarity scores), and we studied the predictive power (Pearson correlation) of each individual feature, whereas Uysal and Croft assessed the predictive power of entire groups of features only.

Uysal and Croft found that features roughly corresponding to our Group 1 (the tweet itself) were the most useful, whereas in our experiments (Section 3.3) only Group 2 (content text similarity), Group 3 (influence, authority, popularity), Group 4 (sender-recipient interaction), and Group 6 (neighbours) had features in the top 10. This difference may be due to the different datasets and learning algorithms that we used. Uysal and Croft used a decision tree classifier, whereas we used logistic regression. Also, we used 122 journalists as recipients, whereas Uysal and Croft used 242 random (but reasonably active) Twitter users. On the other hand, the dataset of Uysal and Croft was smaller (24,200 instances in total) compared to ours (133,000 instances), Uysal and Croft did not examine the effect of the size of the training set, and the tweets of their dataset were not temporally ordered.

Hong et al. [9] use types of features that are similar to the ones we used, but rely on Factorization Machines. We use a much simpler logistic regression classifier, still obtaining very promising results.

Zhang et al. [21] also developed a hybrid personalized global filter (a single filter for all recipients, with recipient-sensitive feature vectors) to predict retweets. They used word embeddings to represent the words of the tweets and a convolutional neural network (CNN) to construct a single embedding for each tweet. The senders and recipients are also represented by (user) embeddings, and their embeddings influence the behaviour of a second version of the CNN that produces an alternative embedding of each tweet, in effect making the second CNN sensitive to the interests of the senders and recipients. The output tweet embeddings of the two versions of the CNN, concatenated with the embeddings of the recipient and sender and the similarity of the scores of the two CNN versions are then used as a feature vector by a logistic regression classifier layer. The work of Zhang et al. is an interesting attempt to avoid manual feature engineering. The embeddings that they use, however, in effect encode information only about the words of the tweet and the previous tweets of the sender and recipient. Our experiments showed that features that consider the influence, authority, and popularity of the sender, the previous interaction between the sender and the recipient, and the neigbours of the recipient are also useful. Their experiments were conducted on a collection of 37,515 incoming tweets from 1,000 random recipients.

A shorter version of this paper has also been published [17].

5 CONCLUSIONS AND FUTURE WORK

We presented a personalized global filter that aims to identify incoming tweets a particular recipient would find interesting enough to retweet. The filter is global in the sense that it is common for all the recipients. It is also personalized in the sense that the incoming tweets are represented as feature vectors that include user-specific features. Thus, the system can produce different predictions per recipient, even for the same incoming tweet, as in personal filters, while still being able to generalize over different users. We experimented with features that examined the content of each tweet, its novelty and its similarity to tweets previously posted or retweeted by the recipient or sender. Furthermore, features describing the network influence and authority of the author and sender, their past interactions and neighbours were used. In experiments with a collection of approximately 130K tweets received by 122 journalists, our system achieved very high accuracy ($F_1 \approx 0.9$) using only 10 features and only 5K training instances. Moreover, although the model was proven to achieve good results with relatively few data, we do not consider this to be a counter-argument to the use of hybrid filter approaches (where their ability to take advantage of more data was one of our main arguments). This is because the number of instances needed to adequately train the model (few thousands) is still large enough in the context of a single, average user and hence, purely personal filters would still be unable to cope with the 'cold start' problem.

Future work could incorporate the features we used (e.g., by turning them into embeddings) in convolutional or recursive neural networks, possibly building upon the work of Zhang et al. [21]. Benchmark datasets are also needed to compare methods proposed by different researchers. The (encoded) dataset we have made available, is a step towards this direction, but the recipients of its tweets were all journalists.

REFERENCES

[1] O. Alonso, C. Carson, D. Gerster, X. Ji, and S. U. Nabar. 2010. Detecting uninteresting content in text streams. In *SIGIR Workshop on Crowdsourcing for Search Evaluation*. Geneva, Switzerland.

[2] O. Alonso, C. C. Marshall, and M. Najork. 2013. Are some tweets more interesting than others? #hardquestion. In *Proceedings of the Symposium on Human-Computer Interaction and Information Retrieval*. Vancouver, Canada, 2.
[3] I. Androutsopoulos, J. Koutsias, K.V. Chandrinos, and C.D. Spyropoulos. 2000. An experimental comparison of Naive Bayesian and keyword-based anti-spam filtering with encrypted personal e-mail messages. In *Proceedings of the SIGIR Conference*. Athens, Greece, 160–167.
[4] J. Benesty, J. Chen, Y. Huang, and I. Cohen. 2009. Pearson correlation coefficient. In *Noise reduction in speech processing*. New York City, USA, 1–4.
[5] I. Chalkidis, I. Androutsopoulos, and A. Michos. 2017. Extracting Contract Elements. In *Proceedings of the 16th International Conference on Artificial Intelligence and Law*. London, UK.
[6] J. Chen, R. Nairn, L. Nelson, M. Bernstein, and E. Chi. 2010. Short and tweet: experiments on recommending content from information streams. In *Proceedings of the Conference on Human Factors in Computing Systems*. Atlanta, USA, 1185–1194.
[7] Y. Duan, L. Jiang, T. Qin, M. Zhou, and H.Y. Shum. 2010. An empirical study on learning to rank of tweets. In *Proceedings of the International Conference on Computational Linguistics*. Beijing, China, 295–303.
[8] K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. Smith. 2011. Part-of-Speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the ACL: Human Language Technologies*. Portland, USA, 42–47.
[9] L. Hong, A. Doumith, and B.D. Davison. 2012. Personalized Retweet Prediction in Twitter. In *Proceedings of the 4th Workshop on Information in Networks*. New York City, NY.
[10] J. Hurlock and M. L. Wilson. 2011. Searching Twitter: Separating the tweet from the chaff. In *Proceedings of the International Conference on Weblogs and Social Media*. Barcelona, Spain, 161–168.
[11] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the International Conference on Neural Information Processing Systems*. Stateline, USA, 3111–3119.
[12] J. Pennington, R. Socher, and C. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Doha, Qatar, 1532–1543.
[13] C. Tan, L. Lee, and B. Pang. 2014. The effect of wording on message propagation: Topic-and-author-controlled natural experiments on Twitter. In *Proceedings of the 52nd Annual Meeting of the ACL*. Baltimore, USA, 175–185.
[14] I. Uysal and W. Bruce Croft. 2011. User oriented tweet ranking: a filtering approach to microblogs. In *Proceedings of the International Conference on Information and Knowledge Management*. Glasgow, Scotland, 2261–2264.
[15] M. Vougioukas. 2014. Development of a system to filter tweets. (2014). BSc thesis, Department of Informatics, Athens University of Economics and Business (http://nlp.cs.aueb.gr/theses/mvougioukas_bsc_thesis.pdf, in Greek).
[16] M. Vougioukas. 2016. A personalised system to predict retweets. (2016). MSc thesis, Department of Informatics, Athens University of Economics and Business (http://nlp.cs.aueb.gr/theses/vougioukas_msc_thesis.pdf, in English).
[17] M. Vougioukas, I. Androutsopoulos, and G. Paliouras. 2017. A personalized global filter to predict retweets. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*. Bratislava, Slovakia.
[18] W. Waldner and J. Vassileva. 2014. Emphasize, don't filter! Displaying recommendations in Twitter timelines. In *Proceedings of the Conference on Recommender Systems*. Foster City, USA, 313–316.
[19] M. Yang and H. Rim. 2014. Identifying interesting Twitter contents using topical analysis. *Expert Systems with Applications* 41, 9 (2014), 4330–4336.
[20] K. Zamani, G. Paliouras, and D. Vogiatzis. 2015. Similarity-based user identification across social networks. In *International workshop on similarity-based pattern recognition*. Copenhagen, Denmark, 171–185.
[21] Q. Zhang, Y. Gong, J. Wu, H. Huang, and X. Huang. 2016. Retweet prediction with attention-based Deep Neural Network. In *Proceedings of the International on Conference on Information and Knowledge Management*. Indianapolis, USA, 75–84.