# Cheap and Cheerful: Trading Speed and Quality for Scalable Social Recommenders

Anne-Marie Kermarrec[1] François Taïani[2] Juan M. Tirado[1]

[1] INRIA Rennes
anne-marie.kermarrec,juan-manuel.tirado@inria.fr
[2] University of Rennes 1 - IRISA - ESIR
francois.taiani@irisa.fr

**Abstract.** Recommending appropriate content and users is a critical feature of on-line social networks. Computing accurate recommendations on very large datasets can however be particularly costly in terms of resources, even on modern parallel and distributed infrastructures. As a result, modern recommenders must generally trade-off quality and computational cost to reach a practical solution. This trade-off has however so far been largely left unexplored by the research community, making it difficult for practitioners to reach informed design decisions. In this paper, we investigate to which extent the additional computing costs of advanced recommendation techniques based on supervised classifiers can be balanced by the gains they bring in terms of quality. In particular, we compare these recommenders against their unsupervised counterparts, which offer lightweight and highly scalable alternatives. We propose a thorough evaluation comparing 11 classifiers against 7 lightweight recommenders on a real Twitter dataset. Additionally, we explore data grouping as a method to reduce computational costs in a distributed setting while improving recommendation quality. We demonstrate how classifiers trained using data grouping can reduce their computing time by 6 while improving recommendations up to 22% when compared with lightweight solutions.

## 1 Introduction

As web and on-line services continuously grow to encompass more facets of our lives, personalization and recommendation are emerging as key technologies to help users exploit the deluge of data they are submitted to. This is particularly true in social-networking applications (Facebook, Google+, Linkedin, Twitter), which receive, store, and, process a continuously growing mass of information produced for tens to hundreds of millions of users daily.

Implementing a recommendation mechanism that works for such a large user base over terabytes of data is a highly challenging task: an ideal solution should be accurate, lightweight, and easily scale to the distributed and cloud environments in which modern recommenders are being deployed [1]. Traditional approaches to user recommendation in social networks have so far heavily relied

on topological metrics to identify new users or items that might be of interest to a user. These approaches, pioneered by Liben-Nowell and Kleinberg [16], can be *unsupervised*, in which a topological metric (e.g. number of common neighbors, length of shortest path) is used directly to predict links in the underlying social graph and thus derive recommendations. These approaches are typically lightweight, and scale well, but can provide sub-optimal recommendations, and tend to depend on the suitability of the chosen metrics for a particular dataset.

In recent years, a second strand has therefore emerged that exploit classifiers developed for machine learning to improve on these earlier approaches [18, 17, 24, 19]. These classifiers often use as inputs the same topological metrics developed for unsupervised learning, and are trained on a part of the social-graph to construct an appropriate prediction model. Due to this training phase, these methods can better adapt to the specifics of individual datasets. They are also able to combine several metrics into one predictor [14], and thus offer a natural path to merge different types of information into a recommender, including topological data, semantic information based on the content consumed and produced by users [18, 24], or geographic information in geolocated social networks (Foursquare, Gowalla) [19].

Unfortunately, training such supervised models can require very large training sets (up to twice as large as the prediction set [17]), and be particularly costly in terms of computation time, even on today's highly distributed, highly parallel infrastructures found in datacenters and cloud providers. Computation costs are in turn a fundamental decision factor [13] used to select practical online recommenders, and has led companies as prominent as Netflix [1] to discard improved, but particularly costly solutions that were difficult to deploy in their target environment (in Netflix's case, Amazon's public cloud).

Almost no information exists nowadays about this fundamental trade-off, balancing training's computation cost on modern infrastructures and the quality of the returned recommendation. This lack of analysis is highly problematic, as it leads to researchers to focus almost exclusively on quality metrics that ignore a decision factor that is key to practitioners. In this paper, we analyze this very trade-off, and present an extensive study that contrast the benefits brought by supervised classifiers against their computational costs on parallel architectures under a wide range of operational assumptions. First, we describe a method that combines topology-based and content-based information to improve the quality of recommendations. Second, we explore the utilization of data grouping methods to reduce the computation time required to train classifiers while improving recommendations. We carry out a thoroughly evaluation using a real dataset extracted from Twitter that demonstrates the benefits our approach can bring to scalable user-recommenders.

This work is structured at follows. In Section 2 we state the problems of user recommendation. Section 3 briefly introduces the related work. Section 4 describes our approach. Section 5 describes the evaluation of this approach and finally we conclude in Section 6.

## 2 Problem statement

The tremendous growth of users data in modern on-line services has made recommendation a key enabling technology in the last few years [4]. This is particularly true in on-line social networks such as Facebook, Twitter, or Weibo, which allow users to maintain an on-line web of social connections, where they produce and consume content. These networks serve up to hundreds of millions of users (for instance Facebook reported 1.15 billion monthly active users in June 2013 [8]), and must select recommendation techniques able to scale to their user base, while being amenable to the highly distributed infrastructures in which these services are typically deployed. The ability to scale and distribute recommendation algorithms has been shown in the past to play a key role in their acceptability: Netflix for instance revealed in 2012 that it had not adopted the winning algorithms of its own one million dollar Netflix prize, in part because of the engineering challenges raised to port the algorithm to their distributed infrastructure (hosted by Amazon) [1].

In this work, we focus on the problem of recommending users to other users. This problem can be compared to the link prediction problem [16] where we try to predict when the user $u$ will create a link with another user $v$. In order to do this, we have to compute a recommendation score ($score(u, v)$) or score function indicating the interest of $u$ to create a link with $v$. How to compute this score, depends on the taken approach. One common approach is to use *unsupervised* models consisting of a generic solution that is oblivious to the distinguishing features of the dataset. A second approach uses supervised models, consisting of classifiers trained with an excerpt of data extracted from the target dataset. Generally, this score is given by a pre-computed model that has been previously trained using the information available in the system. The general approach is to provide the system with a representative number of observations in order to compute an accurate model. Once a score function is chosen, we can compute a matrix of scores among the users in the system and chose the largest scored users as recommendations. Computing the score among all the users is not practical, therefore only a small subset is scored. Normally only a subset of close neighbors are score for each user during the recommendation. Apart from the time cost of computing the score of the neighbors, we have to consider the cost of training the supervised models. This aspect is generally ignored, although it is a major constraint in the design of distributed user-recommenders.

## 3 Related work

The problem of link prediction has been addressed in two separated strands. A first approach follows the seminal work of Liben-Nowell and Kleinberg [16] using metrics based on the network topology [23, 2]. As observed by Yien et al. [25] these metrics only reflect changes in the network topology being oblivious to the creation of links regarding other aspects contained into the users metadata. In this sense, Schifanella et. al [21] find tags to be a good link predictor. However,

in [7] authors find that tags are not very effective for link-prediction in their explored datasets.

A second approach employs methods to combine different features in order to exploit all the available information in the systems. Rowe et al. [18] exploit Twitter semantics using logistic regression. Authors claim the need for topical affinity between users to create links. Although their work has some resemblance with the one we present, their work differs from our approach in the utilization of topics instead of natural language and the analysis of just one classifier. Scellato et al. [19] use supervised learning to predict links in a location-based social network. The authors train a set of classifiers with different location and social-based metrics finding that the combination of these metrics results into an improvement of recommendations. They find that a combination of location and social-based metrics does not significantly improve recommendations compared with only location-based recommenders. Wang et al. [24] present a framework for link-prediction based on an ensemble of classifiers trained with graph features and similarity metrics. They claim a 30% improvement of recommendations when compared with other approaches. However, they ignore the elapsed time in the training process using an exhaustive amount of information during training.

Although the aforementioned works emphasize the combination of different data sources in order to improve recommendations, none of the mentioned works present any conclusion about computational costs. In practice, many of these systems are not scalable and only practical for centralized designs [13]. However, the utilization of methods exploiting different features in user-recommenders has not been analyzed from a computational perspective that may facilitate the design of distributed solutions.

## 4 Proposed approach

User-recommendation is essentially a classification problem where we determine whether a candidate is relevant for a user or not. A $score(u, v)$ function determines the probability of $u$ to establish a link with the candidate $v$ after comparing both users. In socially oriented systems for any user $u$ we find his outgoing edges (social links) $\Gamma(v)$ and part of the content $u$ has consumed or generated. The content can have different formats such as tagged content (e.g. YouTube videos) or natural language text (e.g. posts, tweets). Both of them can be managed in bags of words [18] consisting of a vector containing all the words (or tags) employed by the user. Each user has an associated corpus $C_u$ containing all the employed words, where $P_u(i)$ is the probability of finding the word $i$ into the corpus.

Both $\Gamma(v)$ and $C_v$ can be employed to compute similarity metrics that can be used as score functions ($score(u, v)$). These metrics have been widely used in distributed systems where the user has a partial vision of the network [22, 5] and also in graph-based solutions [16, 3]. The main reason for their utilization is that they are computationally light and summarize relevant features users may

have in common. Table 1 shows the score functions used in this work and their notation.

| | | |
|---|---|---|
| Content-based score functions | | |
| Jensen-Shannon | *js* | $\frac{1}{2}\sum_i P_u(i)\log\frac{P_u(i)}{P_v(i)} + \frac{1}{2}\sum_i P_v(i)\log\frac{P_v(i)}{P_u(i)}$ with $i \in C_u \cap C_v$ |
| Jaccard | *jacc* | $\lvert C_u \cap C_v\rvert / \lvert C_u \cup C_v\rvert$ |
| Cosine | *cos* | $1 - \frac{\sum_i w_u(i)w_v(i)}{\sqrt{\sum_i w_u(i)^2}\sqrt{\sum_i w_v(i)^2}}, i \in C_u \cap C_v$ |
| Adamic-Adar | *aa* | $\sum_{i \in C_u \cap C_v} 1/\log P_u(i)$ |
| Graph-based score functions | | |
| Jaccard | *jaccf* | $\lvert\Gamma(u) \cap \Gamma(v)\rvert / \lvert\Gamma(u) \cup \Gamma(v)\rvert$ |
| Adamic-Adar | *aaf* | $\sum_{i \in \Gamma(u)\cap\Gamma(v)} 1/\log\lvert\Gamma(i)\rvert$ |
| Preferential attachment | *pa* | $\lvert\Gamma(u)\rvert\lvert\Gamma(v)\rvert$ |

Table 1: $score(u, v)$ functions based on content and graph information.

The Jensen-Shannon divergence measures the distance between two corpuses using the probability distribution of the words used by each user. The Jaccard coefficient is a common metric that measures the probability of sharing items between users. Cosine distance considers the elements and their occurrences as the dimensions of two vectors. Adamic-Adar weights rare common features [16]. Finally, preferential attachment considers the probability of connecting two users proportionally to their connectivity degree. For the Jaccard and Adamic-Adar metrics we present a content-based and a social graph based version.

The aforementioned score functions can be directly used as unsupervised score-based classifiers to compute the probability of $u$ and $v$ to become connected or not. However, in some scenarios computing recommendations remains a challenge. For example, cold-start recommendations [20] will probably fail as there is no available data. In other scenarios the score can be biased. For example, if $u$ and $v$ have similar content and a low number of common links, probably $v$ is not relevant to $u$. They share similar topics but are distant neighbors. And similarly, a low content score wit a great social similarity indicates close users although they consume different contents. Combining both approaches (content-based and social-based) at the same time can improve recommendations in scenarios where only one approach may be insufficient.

### 4.1 Supervised multi-score recommenders

There is an extensive literature in supervised classification algorithms [15] that take advantage of different statistic features. Supervised classifiers have to be trained with a given set of observations (training dataset) each one containing

a set of features and the class belonging to. Depending on the classifier and the training dataset, the classifier will come up with a different classification model.

In our approach, we propose to use a training dataset with entries containing $\{s_1, s_2, ..., s_n, c\}$ where $s_i$ is a score function (Table 1) and $c$ indicates whether users $u$ and $v$ are connected (1 if connected, 0 otherwise). Ideally, for every user $u$ in the social graph, we can create a training dataset and then train the corresponding classifier. However, this is not a scalable solution as we would need to compute as many classifiers as users in the system. Additionally, computing a single model for the whole dataset requires identifying a representative enough sample which is a difficult task. In order to cope with this problem, we split the graph into manageable groups and compute classifiers for these groups. Intuitively, groups composed of users with similar features should have similar classification models. This permits to train personalized recommendation models for samples of users. Additionally, by splitting the problem we can consider the parallelization through distribution using paradigms such as Map-reduce.

**Algorithm 1** Supervised score-based model training

**for** each group $g$ **do**
  $T \leftarrow \{\}$ training set
  $U$ random sample of users belonging to $g$
  //Fill the training dataset
  **for** $u \in U$ **do**
    $N \leftarrow neighborsSelection(u, d)$
    **for** $n \in N$ **do**
      Compute each similarity metric $i$
      $s_i \leftarrow score_i(u, n)$
      $c \leftarrow connected(u, n)$
      Add $[s_1, s_2...s_n, c]$ to $T$
    **end for**
  **end for**
  //Find the best classification model
  $B_i \leftarrow \{\}$
  **for** each classifier $i$ **do**
    $B_i^c \leftarrow \{\}$
    **for** each configuration $c$ **do**
      Train $M_c$ using configuration $c$ and cross-fold validarion over $T$
      Add $\text{AUC}_{M_c}$ to $B_i^c$
    **end for**
    $M_i \leftarrow$ model with largest value from $B_i^c$
    Save $M_i$, discard the other models
  **end for**
  $M_g \leftarrow$ model with largest AUC from $B_i$
**end for**

Algorithm 1 describes the steps carried out to train and find the most suitable model for each group of users $g$. First, we define the training set $T$ for each group

$g$. For each $g$, we select a sample of users $U$ large enough to be representative but small enough to be computationally feasible. The users are chosen using the $neighborsSelection(u, d)$ function as explained in Section 4.2. For each of the users in $U$ we compute the score functions and add them to the training set $T$. Once we have all the training sets, we can compute the best classification model. Considering the most adequate classifier for a given social graph a priori is a difficult task. Different classifiers show different performance depending on the incoming training set. We propose to simultaneously train several classifiers using cross-fold validation over $T$. For each classifier we train models $M_c$ for a set of pre-determined configurations (if the classifier accepts additional configuration), compute the obtained AUC (Area Under the ROC Curve) for $M_c$. The AUC measures the predictive power of the model between 0 and 1. Values larger than 0.5 indicate better performance than a random classifier. The model with the largest AUC will finally be selected as the classifier for the group ($M_g$).

### 4.2 Training set users selection

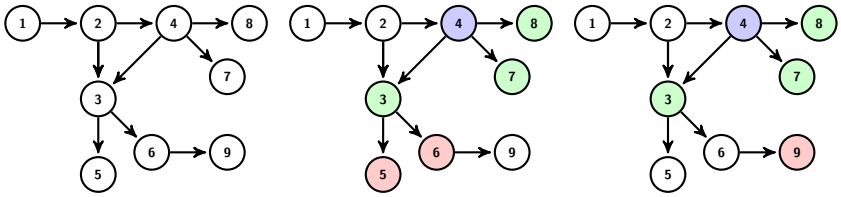
The training set $T$ described in Algorithm 1 must contain a proportional ratio of observations belonging to both classes (connected and non-connected) in order to get an accurate classifier. We select a random sample of users $U$ from group $g$. Then for each user $u$ in $U$ we compute the features corresponding to a connected and a non-connected user. Selecting a connected user we just have to select one $v$ belonging to $\Gamma(u)$ and compute the different $score(u, v)$ functions. However, the remaining users in the graph could be considered as non-connected examples. We propose a social distance approach to determine which users to consider as non-connected.

We define the social distance $d$ as the minimal number of links $u$ has to traverse in order to find user $v$. Previous works observe that most of the new links in social networks are established for small values of $d$ [25]. We use the social distance to determine when a user shall be considered as an example of connected or non-connected class in the training set. For $d = 1$ we consider the users to be connected as they are currently neighbors. Then for $d > 1$ we consider non-connected samples. According to this assumption, for large values of $d$ a classifier must find easier to distinguish between connected and non-connected users during training. Figure 1 describes how the $neighborsSelection(u, d)$ function works. In Figure 1a we have the original directed social graph. In Figure 1b we use $d = 2$ selecting users 3, 7 and 8 (green) as connected examples with 5 and 6 as non-connected taking 4 as the origin. Similarly, in Figure 1c we use $d = 3$ being 9 the only non-connected candidate.

## 5 Evaluation

We evaluate our approach using a Twitter dataset extracted using the public Twitter API [3]. We have crawled Twitter's social graph for users in the London

[3] `http://dev.twitter.com/docs/api/1.1`




(a) Social graph (b) $d = 2$ (c) $d = 3$

Fig. 1: Example of training set users selection using $d = 2$ and $d = 3$. Vertex 4 indicates the starting user. Vertices 3, 7 and 8 indicate examples of connected users. With $d = 2$ vertices 5 and 6 are non-connected vertices. With $d = 3$ the non-connected vertex is 9.

area extracting their tweets and list of followings (users they follow) before July $28^{th}$ 2013. Our dataset accounts for 106,385 users with 11,111,386 following links and a total of 21 million tweets. The median of the distribution is 94 followings with 80% of the users having less than 265 followings.

We aim to understand how a combination of graph and user content information can improve user recommendations and how we can reduce the computational overhead in order to facilitate the scalability of this process. We use the score functions defined in Table 1 in order to compare two users in terms of social and content similarities. We use the tweets to compute the content-based score functions. Tweets use natural language that may reduce the amount of information we can extract from them. For that reason, we first remove stop words and punctuation symbols (except hashtags).

We use a set of representative classifiers available in the R Caret package [11]. This package offers a unified interface for a large number of classifiers, simplifying the implementation. Table 2 enumerates the classifiers we have used for our experiments. We choose these classifiers in order to have a representative set with various classifiers that may get some benefit from our approach. For each training we use cross-fold validation with 10 folds. Then we select a sample of random users with a variable social distance $d$ and compute the AUC to measure the quality of recommendations. In order to compare the computational cost of training we show the average elapsed time for 5 executions. All the experiments are carried out in a non-fully dedicated 8 Intel Xeon cores machine with 32 GBytes of memory.

### 5.1 Single-score recommenders

In this section, we show the quality of recommendations simply using score functions as recommenders. For clarity, we analyze the recommendation power depending on the nature of the score (graph-based or content-based), the number of user followings and the social distance $d$ of the users in the evaluation set. The results showed in Figure 2 indicate that increasing the social distance $d$ improves the performance of recommenders. However, graph-based recommenders

| Method | Abbreviated name |
|---|---|
| Bagging | *bagFDA* |
| Gradient boosted models [9] | *gbm,blackboost* |
| Decision tree | *C5.0Tree* |
| Random forests [6] | *parRF* |
| K-nearest neighbors | *knn* |
| Multivariate adaptive regression splines [10] | *earth* |
| Logistic regression | *glm,glmnet* |
| General additive models | *gam* |
| Support Vector Machine | *svmLinear* |

Table 2: Summary of employed classifiers used in the evaluation and their abbreviated names.

are more accurate than content-based. In particular, we observe that the AUC increases for users with a larger number of followings. This could be due to the cold-start effect that limits the amount of available information. However, in the case of content-based recommenders we observe that the number of followings do not substantially modify the recommendations.

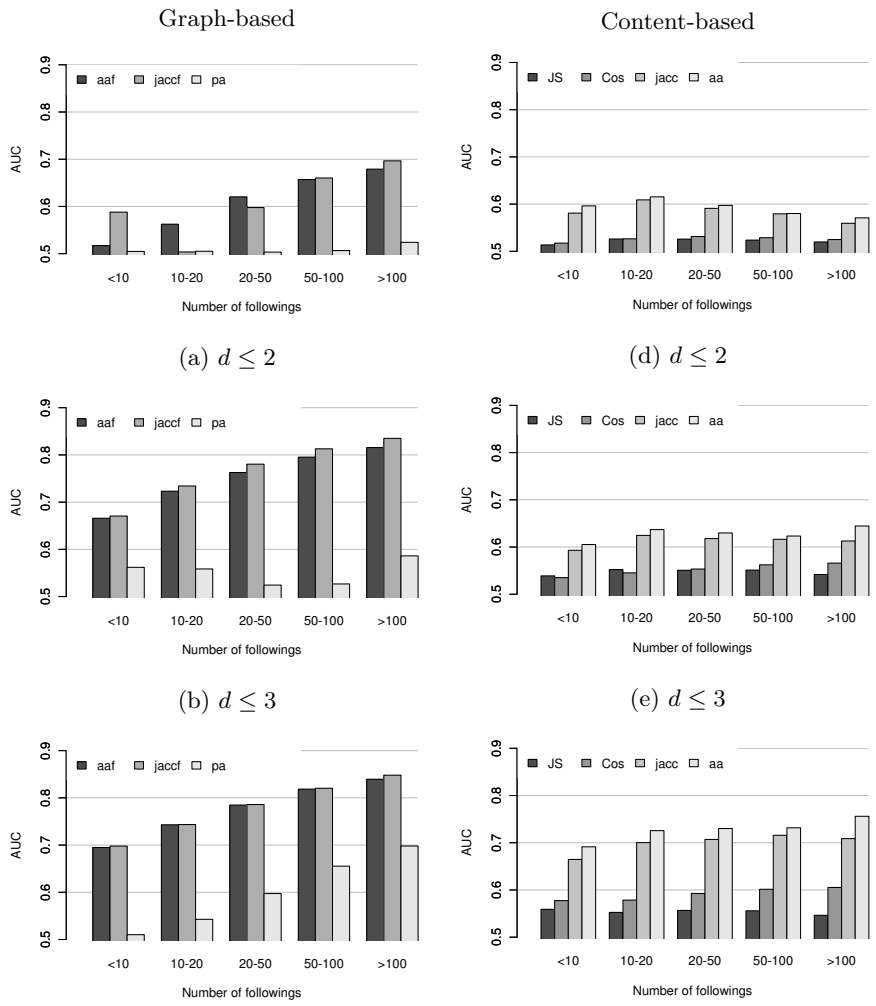
### 5.2 Supervised multi-score recommenders

After analyzing the recommendations obtained using score functions, we explore the recommendations obtained with multi-score recommenders. We use the same evaluation set employed in the previous section with $d \leq 2$. First we only combine score functions from graph-based (Figure 3a) and content-based (Figure 3b) score functions. Then we combine both in order to check how by combining data sources we can improve recommendations (Figure 3c).

For graph-based multi-score recommenders (Figure 3a) we observe a significant improvement for the users with less than 100 followings. This improvement is particularly relevant for users with less than 10 followings achieving a 22% improvement (0.71 AUC compared with graph-based single-score that only achieved 0.58). However, we do not observe relevant improvements for content-based multi-score recommenders (Figure 3b) compared with the single-score version. Finally, the combination of all the scores (Figure 3c) significantly improves the recommendations of some classifiers such as *glm* or *svmLinear*. In the other cases there is an improvement of the recommendations, although it is not very significant.

### 5.3 Train set grouping

In the previous section we show how using multi-score classifiers improves recommendations. However, in order to facilitate the deployment of a distributed solution we have to consider the computation cost of training classifiers. The elapsed time training a classifier depends on the amount of data and the classifier itself. Additionally, as described in Algorithm 1 our approach considers




(a) $d \leq 2$ (d) $d \leq 2$

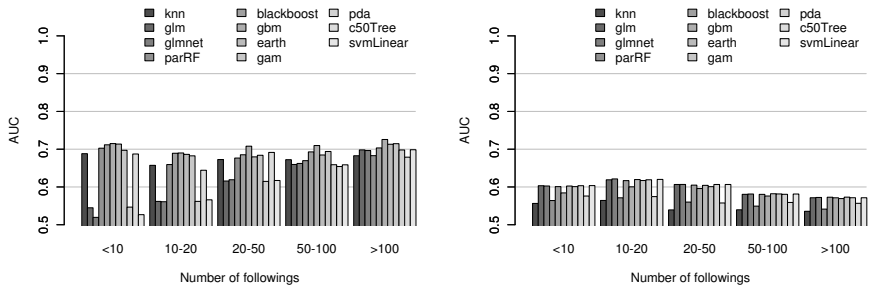
(b) $d \leq 3$ (e) $d \leq 3$

(c) $d \leq 4$ (f) $d \leq 4$

Fig. 2: AUC according to the number of followings and the social distance for content-based similarity metrics.

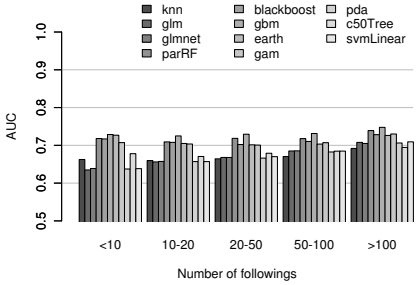
the training of several models in order to find the most accurate model. This operation may require a significant amount of time. In order to reduce this time while keeping the quality of recommendations, we split users into groups.

In our experiments we split the dataset into five equally-sized groups depending on the number of outgoing edges. The reason behind this partition is to reduce the diversity of features found in each training set. Intuitively, users with the same number of outgoing links may have similar profiles, and therefore it would be easier to find a model to classify them. We assume that the




(a) Graph-based multi-score recommenders

(b) Content-based multi-score recommenders




(c) Graph and content-based multi-score recommenders

Fig. 3: AUC for multi-score recommenders using graph-based scores, content-based scores and both kind of scores.

information regarding each user (followings and content) is fully available when computing the scores. In Figure 4 we plot the elapsed average time for training and the AUC per classifier using a multi-score recommender combining graph and content-based scores. We classify the same set of users employed in the previous section with and without grouping (left and right highlighted areas respectively).

We observe an increment of the AUC for all the classifiers with a significant time reduction after grouping. In some cases like *parRF* the AUC slightly improves while the training is 6 times faster. In other cases like the *earth* classifier, the AUC increment is more significant than the saved time. This experiment shows how partition permits to reduce the training time while not affecting the quality of recommendations.

### 5.4 Discussion

There are many aspects to be evaluated in a recommendation system. We know the evaluation presented in this paper is not complete. However, we think that many of the results presented in this work are promising and open the door for

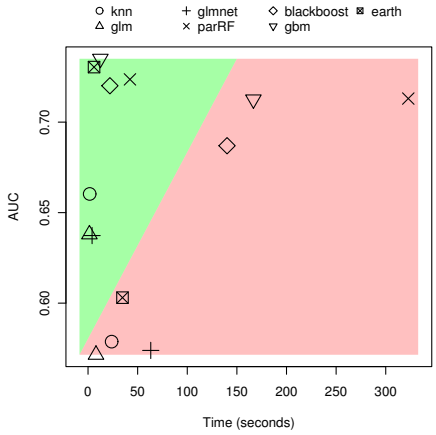



Fig. 4: Comparison of AUC and time used in model training. The right and left areas indicate the results not using and using groups respectively.

new ideas in the development of scalable recommendation systems that can combine multiple sources of information. Our experiments demonstrate how training classifiers using classic user comparison metrics can improve recommendations. We observe that there is a significant improvement when dealing with users offering small amounts of data. Furthermore, we demonstrate how classifiers trained with metrics based on different sources of information we can get 22% better recommendations than the best value obtained simply using these metrics. The combination of graph and content-based slightly improves recommendations. However, this result may vary in other datasets.

Group partitioning demonstrates to be a good approach to reduce the average elapsed time in classifiers training. Additionally, we observe how recommendation improves for all the evaluated classifiers after grouping. This result is particularly promising in order to develop distributed recommender systems using supervised classifiers. Grouping data with common features improves recommendations. This makes possible to reduce the amount of training data, and therefore the elapsed training time. In this work, we only provide one group partitioning strategy based on the number of outgoing links of the user. A large number of partition techniques based on topology features can be explored [12]. However, our main goal is to demonstrate the importance of group partitioning in terms of recommendations and time. Exploring other grouping techniques remains for future work. Finally, the reduction in the time needed to train the models makes possible to compute a larger number of models. And facilitates the parallelization of the recommendation process. Our experiments indicate that the *gbm* (Gradient Boosted Modelling) classifier gets the best results in almost every scenario. However, this result can differ depending on the dataset.

## 6 Conclusions

In this work we explore the design of scalable solutions for social recommenders. First, we propose the utilization of supervised classification methods for user recommendation in social networks. We describe a method that combines topology-based and content-based similarity metrics to improve the quality of recommendations. Second, we explore the utilization of data grouping methods to reduce the computation time required to train classifiers and make easier the deployment of distributed solutions. We carry out a thoroughly evaluation using a real dataset extracted from Twitter that demonstrates the benefits of our approach. In particular, we find that our solution improves the quality of recommendations by 22% compared with unsupervised solutions. Additionally, we observe that data grouping permits to speedup the training of classifiers by 6.

Our work shows promising results and opens several directions in the development of scalable social recommenders. We plan to extend our study about the effects of data grouping in the quality of recommendations and how it facilitates the deployment of scalable solutions. Additionally, we will extend our evaluation to other datasets.

## Acknowledgments

This work was partially funded by the French ANR project SocioPlug (ANR-13-INFR-0003), and by the DeSceNt project granted by the Labex CominLabs excellence laboratory (ANR- 10-LABX-07-01).

## References

1. X. Amatriain and J. Basilico. Netflix recommendations: Beyond the 5 stars. `http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html`, April 6 2012. (accessed 20/9/2013).
2. L. Backstrom and J. Leskovec. Supervised random walks: predicting and recommending links in social networks. WSDM '11, pages 635–644, 2011.
3. A. L. Barabasi, J. De, P. Lettres, L. Al, N. Cimento, H. Jeong, H. Jeong, Z. Neda, Z. Neda, and A. L. Barabasi. Measuring preferential attachment in evolving networks. *Europhysics Letters*, 61(61):567–572, 2003.
4. J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46(0):109 – 132, 2013.
5. A. Boutet, D. Frey, R. Guerraoui, A. Jégou, and A.-M. Kermarrec. WhatsUp Decentralized Instant News Recommender. In *IPDPS 2013*, May 2013.
6. L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
7. M. J. Brzozowski and D. M. Romero. Who should i follow? recommending people in directed social networks. In *ICWSM*, 2011.
8. Facebook Inc. Key facts. `https://newsroom.fb.com/Key-Facts`, 2013. (accessed 2 Oct. 2013).

9. Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997.
10. J. H. Friedman. Multivariate Adaptive Regression Splines. *The Annals of Statistics*, 19(1):1–67, 1991.
11. M. K. C. from Jed Wing, S. Weston, A. Williams, C. Keefer, A. Engelhardt, and T. Cooper. *caret: Classification and Regression Training*, 2013. R package version 5.16-24.
12. J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin. Powergraph: Distributed graph-parallel computation on natural graphs. In *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*, OSDI'12, pages 17–30, Berkeley, CA, USA, 2012. USENIX Association.
13. P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh. Wtf: the who to follow service at twitter. In *WWW*, pages 505–514, 2013.
14. M. Hasan and M. Zaki. A survey of link prediction in social networks. In C. C. Aggarwal, editor, *Social Network Data Analytics*, pages 243–275. Springer US, 2011.
15. S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, pages 3–24, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press.
16. D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. CIKM '03, pages 556–559. ACM, 2003.
17. R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla. New perspectives and methods in link prediction. KDD '10, pages 243–252, New York, NY, USA, 2010. ACM.
18. M. Rowe, M. Stankovic, and H. Alani. Who will follow whom? exploiting semantics for link prediction in attention-information networks. ISWC'12, pages 476–491, Berlin, Heidelberg, 2012. Springer-Verlag.
19. S. Scellato, A. Noulas, and C. Mascolo. Exploiting place features in link prediction on location-based social networks. KDD '11, pages 1046–1054, New York, NY, USA, 2011. ACM.
20. A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. SIGIR '02, pages 253–260, New York, NY, USA, 2002. ACM.
21. R. Schifanella, A. Barrat, C. Cattuto, B. Markines, and F. Menczer. Folks in folksonomies: social link prediction from shared metadata. WSDM '10, pages 271–280, 2010.
22. J. M. Tirado, D. Higuero, F. Isaila, J. Carretero, and A. Iamnitchi. Affinity p2p: A self-organizing content-based locality-aware collaborative peer-to-peer network. *Comput. Netw.*, 54(12):2056–2070, Aug. 2010.
23. J. Valverde-Rebaza and A. de Andrade Lopes. Structural link prediction using community information on twitter. CASoN'12, pages 132–137, 2012.
24. C. Wang, V. Satuluri, and S. Parthasarathy. Local probabilistic models for link prediction. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 322–331. IEEE, 2007.
25. D. Yin, L. Hong, and B. D. Davison. Structural link analysis and prediction in microblogs. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 1163–1168, New York, NY, USA, 2011. ACM.