A RECOMMENDATION ALGORITHM FOR PERSONALIZED ONLINE NEWS BASED ON COLLECTIVE INTELLIGENCE AND CONTENT

Giovanni Giuffrida
Department of Social Sciences, University of Catania, Catania, Italy

Calogero G. Zarba
Neodata Intelligence s.r.l., Catania, Italy

Keywords: Recommender systems, Text mining, Data mining.

Abstract: We present a recommendation algorithm for online news based on collective intelligence and content. When a user asks for personalized news, our algorithm recommends news articles that (i) are popular among the members of the online community (the collective intelligence part), and (ii) are similar in content to the news articles the user has read in the past (the content part).
Our algorithm computes its recomendations based on the collective behavior of the online users, as well as on the feedback the users provide to the algorithm's recommendations. The users' feedback can moreover be used to measure the effectiveness of our recomendation algorithm in terms of the information retrieval concepts of precision and recall.
The cornerstone of our recommendation algorithm is a basic relevance algorithm that computes how relevant a news article is to a given user. This basic relevance algorithm can be optimized in order to obtain a faster online response at the cost of minimal offline computations. Moreover, it can be turned into an approximated algorithm for an even faster online response.

1 INTRODUCTION

Online newspapers are blooming because the online medium offers advantages that are simply not available in the printed medium. Online news can be delivered in real time. Online news can be paid individually with micro payments. Online news can be personalized.

With personalization, a user visiting an online newspaper website can go to a personalized webpage. Different users will see different versions of the personalized webpage. The content of the personalized webpage is generated by using a recommender system, which uses a recommendation algorithm (or a combination thereof) in order to select articles specifically targeted to the interests of the user.

In this paper, we present a recommendation algorithm based on collective intelligence (Segaran, 2007) and content (Pazzani and Billsus, 2007). When a user asks for personalized news, we recommend those articles that (i) are popular among the members of the online community (the collective intelligence part), and (ii) are similar in content to the articles the user has read in the past (the content part).

After our recommendation algorithm computes the recommendations, the user may provide feedback by stating, for each recommended article, whether she likes the article or not. This feedback allows us to measure the effectiveness of our recommendation algorithm in terms of the information retrieval concepts of precision and recall.

More in detail, our recommendation algorithm works as follows. For each candidate article, we keep an up-to-date popularity score, which measures the current level of interest of the online community to the article. The popularity score is computed based on the collective behavior of the online users, that is, by keeping track of which articles are read by the online users in the recent past.

Each time a user asks for personalized news, we use a basic relevance algorithm that computes, for each candidate article, a relevance score that measures

Giuffrida G. and G. Zarba C..
A RECOMMENDATION ALGORITHM FOR PERSONALIZED ONLINE NEWS BASED ON COLLECTIVE INTELLIGENCE AND CONTENT.
DOI: 10.5220/0003115401890194
In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence* (ICAART-2011), pages 189-194
ISBN: 978-989-8425-40-9
Copyright © 2011 SCITEPRESS (Science and Technology Publications, Lda.)

how much the article is relevant to the requesting user. The relevance score is computed based on the behavior of the requesting user, that is, by keeping track of the articles read by the user, as well as of the feedback the user provides to the recommendations received.

The popularity scores and the relevance scores are then combined, obtaining a final score for each candidate article. Those articles with the highest final scores are recommended to the user.

The basic relevance algorithm works as follows. For each user, we keep track of the set of articles she explicitely likes. When a user asks for personalized news, we construct an appropriate data structure which, based on the set of articles explicitely liked by the user, is able to model the current interests of the user. This data structure is compared with the content of each candidate article, for which a relevance score is computed.

The basic relevance algorithm gives very high quality results, but its time complexity is high, since it is proportional to the product of the number of candidate articles with the average number of distinct words contained in an article. The practical consequence is that the basic relevance algorithm is not scalable: it is slow when the number of candidate articles is high.

To address this scalability problem, we devised an optimized relevance algorithm that gives the same results of the basic relevance algorithm, but has a faster online response at the cost of minimal offline computations. The (online) time complexity of the optimized relevance algorithm is proportional to the product of the number of candidate articles with the number of articles explicitely liked by the user.

The optimized recommendation algorithm has a faster online response than the basic recommendation algorithm. However, in practice the optimized relevance algorithm is still not scalable: it is still slow when the number of candidate articles is high.

To obtain a scalable algorithm, we show how the optimized relevance algorithm can be turned into an approximated relevance algorithm that is much faster and needs the same minimal offline computations. The (online) time complexity of the approximated relevance algorithm is proportional to the number of articles explicitely liked by the user. Consequently, the approximated relevance algorithm is scalable, since its response time is not sensitive to the number of candidate articles.

We have implemented a prototype of our recommender system using the Java programming language. Such prototype is currently fed by specific tags posted on one of the largest European online newspapers.

2 RELATED WORK

Several recommender systems for news articles have been developed. NewsWeeder (Lang, 1995) is a content-based recommender system for newsgroup articles, which allows the user to rate articles on a scale from 1 to 5. Using such ratings, NewsWeeder builds a user model for predicting the rating of the user on unseen articles. The unseen articles with the highest predicted rating are recommended to the user. The model is built by applying a combination of naive Bayes classifiers with linear regression. NewsWeeder needs to rebuild the user model every night with an offline computation.

Krakatoa Chronicles (Bharat et al., 1998) is a content-based recommender system for news articles delivered as a Java applet. Based on the content of the articles and past user ratings, Krakatoa Chronicles computes, for each unseen article, a user score and a community score. A weighted average of the user score and community score produces a recommendation score. The unseen articles with the highest recommendation scores are recommended to the user. The community score of an article is the average of all the user scores of the article. When the number of users is in the order of millions, as is common the case, Krakatoa's computation of the community score is computationally expensive.

PersoNews (Banos et al., 2006) is a news reader which filters unseen articles using a naive Bayes classifier. The classifier, which can be trained by user feedback, labels articles as "interesting" or "not interesting". Interesting articles are recommended to the user, while not interesting articles are not. PersoNews also allows the user to monitor topics, which are modelled as sets of keywords. An article belongs to a topic if it contains one of the keywords of the topic.

Hermes (Borsje et al., 2008) is an ontology-based news recommender system. A complex ontology classifies articles in concept categories. The system recommends to the user those articles belonging to the concept categories selected by the user.

Google News (Das et al., 2007) is a news aggregator and recommender system. By entering a list of keywords, the user can retrieve a set of articles matching the keywords. Furthermore, the user can asks for personalized news, which are computed using algorithms based on collaborative filtering.

3 ARTICLES

We denote with $\mathcal{A}$ the set of all articles. We assume that $\mathcal{A}$ is finite.

We model time with the set $\mathbb{N}$. Given an article a, we denote with $date(a)$ the instant of time in which a is published. It $t \geq date(a)$, then we let $age_t(a) = t - date(a)$.

Let t be an instant of time. We denote with $\mathcal{A}_t$ the set of articles published at the instant of time t or before. Formally, $\mathcal{A}_t = \{a \in \mathcal{A} \mid date(a) \leq t\}$.

3.1 Content of Articles

We denote with $\mathcal{W}$ the set of all words. We assume that $\mathcal{W}$ is finite.

Let w be a word, and let a be an article. The TERM FREQUENCY $tf(w,a)$ of w in a is the number of times w occurs in a.

Let $A \subseteq \mathcal{A}$ be a set of articles, and let w be a word. The INVERSE DOCUMENT FREQUENCY $idf(w,A)$ of w with respect to A is

$$idf(w,A) = \log \frac{|A|}{1+|A_w|} ,$$

where A_w is the set of articles in A such that w occurs in a.

Let w be a word, let a be an article, and let A be a set of articles. The TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY $tfidf(w,a,A)$ of w in a with respect to A is

$$tfidf(w,a,A) = tf(w,a)\, idf(w,A) .$$

Let a be an article, and let t be an instant of time. We model the content of the article a relative to the instant of time t with the function $f_a : \mathcal{W} \to [0,1]$ defined by letting

$$f'_a(w) = tfidf(w,a,\mathcal{A}_{date(a)})$$

and

$$f_a(w) = \frac{f'_a(w)}{|f'_a|} ,$$

where

$$|f'_a| = \sqrt{\sum_{w' \in \mathcal{W}} \left[f'_a(w')\right]^2} .$$

3.2 Similarity between Articles

We model the similarity between articles with the function $\sigma : \mathcal{A} \times \mathcal{A} \to [0,1]$ defined by

$$\sigma(a,b) = f_a \times f_b = \sum_{w \in \mathcal{W}} f_a(w) f_b(w) .$$

Notice that $0 \leq \sigma(a,b) \leq 1$. Thus, when $\sigma(a,b)$ is close to 1, articles a and b are similar. When instead $\sigma(a,b)$ is close to 0, articles a and b are not similar.

4 USERS

In this section we present a model for describing the behavior of online users, and for determining a set of articles that are liked by a single user. The set of articles liked by a single user will be then used by our recommendation algorithm in order to compute the recommendations.

We denote with $\mathcal{U}$ the set of all users. We assume that $\mathcal{U}$ is finite.

Users may perform several actions while browsing online. In particular, they may read articles and they may provide an explicit feedback about the articles they have read or that have been suggested to them by a recommender system.

4.1 Reading Actions

At each instant of time t, a user may read an article a. More formally, there is a function $reading : \mathcal{U} \times \mathcal{A} \times \mathbb{N} \to \{0,1\}$ such that $reading(u,a,t) = 1$ if and only if the user u reads the article a at the instant of time t.

4.2 Feedback Actions

At each instant of time t, a user u may provide a feedback to an article a. More formally, there is a function $feedback : \mathcal{U} \times \mathcal{A} \times \mathbb{N} \to \{-1,0,1\}$ such that:

- if $feedback(u,a,t) = -1$ then, at the instant of time t, the user u states that she does not like article a;
- if $feedback(u,a,t) = 0$ then, at the instant of time t, the user u does not state any preference about article a;
- if $feedback(u,a,t) = 1$ then, at the instant of time t, the user u states that she likes article a.

4.3 Liking Set

From the reading actions and the feedback actions, it is possible to define, for each user u, the set $liking_{u,t}$ of all articles that are explicitely liked by u at the instant of time t.

Formally, we have $a \in liking_{u,t}$ if and only if there exists an instant of time $t' \leq t$ such that:

- $feedback(u,a,t') = 1$ or $reading(u,a,t') = 1$;
- $feedback(u,a,t'') \neq -1$, for all instants of time t'' such that $t' < t'' \leq t$.

4.4 Profile

The liking sets can be used in order to construct profiles of users. These profiles are used by our recommendation algorithm in order to compute the recommendations.

Let u be a user, and let t be an instant of time. We model the profile of the user u relative to the instant of time t with the function $g_{u,t} : \mathcal{W} \to [0,1]$ defined by letting

$$g'_{u,t}(w) = \sum_{a \in liking_{u,t}} f_a(w),$$

and

$$g_{u,t}(w) = \frac{g'_{u,t}(w)}{|g'_{u,t}|},$$

where

$$|g'_{u,t}| = \sqrt{\sum_{w' \in \mathcal{W}} \left[g'_{u,t}(w')\right]^2}.$$

5 RECOMMENDER SYSTEMS

Intuitively, at each instant of time t, a recommender system recommends to a user u a set of articles in $\mathcal{A}$. Formally, we model a recommender system as a computable function $\mathbf{RS} : \mathcal{U} \times \mathbb{N} \to 2^{\mathcal{A}}$ such that $\mathbf{RS}(u,t) \subseteq \mathcal{A}_t$.

In this section, we describe one such computable function, that is, we describe our recommendation algorithm. Furthermore, we show how the effectiveness of a recommender system can be measured in terms of the information retrieval concepts of precision and recall.

5.1 The Algorithm

At each instant of time t, our recommendation algorithm recommends articles from a set $\mathcal{C}_t \subseteq \mathcal{A}_t$ of candidate articles.

Given a user u and an instant of time t, our recommendation algorithm performs the following steps:

Popularity Step. For each candidate article $a \in \mathcal{C}_t$, compute $popularity_t(a)$.

Relevance Step. For each candidate article $a \in \mathcal{C}_t$, compute $relevance_t(a,u)$.

Score Step. Normalize all $popularity_t(a)$, so that they all lie from 0 to 1. Normalize all $relevance_t(a,u)$, so that they all lie from 0 to 1. For fixed weights p and r, compute $score_t(a,u) = p \times popularity_t(a) + q \times relevance_t(a,u)$, for each candidate article $a \in \mathcal{C}_t$.

Recommendation Step. Recommend to user u the top n candidate articles in $\mathcal{C}_t$ with the highest score.

5.2 Precision

To measure the precision of a recommender system $\mathbf{RS}$, we perform the following test repeatedly. At some instant of time t, for a user u, we compute $\mathbf{RS}(u,t)$. Then, at the instant of time $t+1$ we monitor the feedback given by the user u, that is, we ask for $feedback(u,a,t+1)$, for each article $a \in \mathbf{RS}(u,t)$. Formally, the precision of the recommender system $\mathbf{RS}$ with respect to the user u and relative to the instant of time t is given by

$$precision(\mathbf{RS},u,t) = \frac{|\mathbf{RS}(u,t) \cap liking_{u,t+1}|}{|\mathbf{RS}(u,t)|}.$$

5.3 Recall

To measure the recall of a recommender system $\mathbf{RS}$ we perform the following test repeatedly. At some instant of time t, for a user u, we compute $\mathbf{RS}(u,t)$, and we compare the result obtained with the set $liking_{u,t}$. Formally, the recall of a recommender system $\mathbf{RS}$ with respect to the user u and relative to the instant of time t is given by

$$recall(\mathbf{RS},u,t) = \frac{|\mathbf{RS}(u,t) \cap liking_{u,t}|}{|\, liking_{u,t}\,|}.$$

6 POPULARITY

Given an article a and an instant of time t, we want to compute a number $popularity_t(a)$ that measures the level of interest of the community to article a at the instant of time t.

We have developed three measures of popularity: decaying popularities, discounted popularities, and sliding window popularities.

6.1 Decaying Popularities

A reading event e consists of a user u reading an article a at an instant of time t. The instant of time in which the event e occurs is denoted with $date(e)$. If $t \geq date(e)$, then we let $age_t(e) = t - date(e)$.

We denote with $\mathcal{R}_{a,t}$ the set of all events e in which a user u reads the article a at instant of time t or before.

We assume that, for each instant of time t, a user can read at most one article. It follows that $\mathcal{R}_{a,t}$ is a finite set. We also assume that an article may be read only after it is published, that is, $\mathcal{R}_{a,t} = \emptyset$ if $t \leq date(a)$.

The decaying popularity $\pi_{h,t}(a)$ of an article a at the instant of time t with respect to a decay half-time $h > 0$ is defined by

$$\pi_{h,t}(a) = \sum_{e \in \mathcal{R}_{a,t}} \left(\frac{1}{2}\right)^{age_t(e)/h} .$$

6.2 Discounted Popularities

The discounting popularity $\pi_t^\infty(a)$ of an article a at the instant of time t is defined by

$$\pi_t^\infty(a) = \frac{|\mathcal{R}_{a,t}|}{age_t(a)} .$$

6.3 Sliding Window Popularities

The sliding window popularity $\pi_t^\delta(a)$ of an article a at the instant of time t with respect to a sliding window $\delta > 0$ is defined by

$$\pi_t^\delta(a) = \frac{|\mathcal{R}_{a,t} \setminus \mathcal{R}_{a,t-\delta}|}{\min(\delta, age_t(a))} .$$

7 RELEVANCE

Given a user u and an instant of time t, we want to devise an algorithm that computes a number $relevance_t(a,u)$, for each candidate article $a \in \mathcal{C}_t$. The number $relevance_t(a,u)$ measures how relevant the candidate article a is to user u at the instant of time t.

We compute $relevance_t(a,u)$ using the age $age_t(a)$ of article a at the instant of time t, the function f_a representing the content of article a, and the function $g_{u,t}$ representing the interests of user u at the instant of time t.

We have developed three measures of relevance: basic relevance, optimized relevance, and approximated relevance.

The optimized relevance is equal, modulo a positive factor that depends only on u and t, to the basic relevance. The optimized relevance can be computed faster than the basic relevance at the cost of some offline computations. The approximated relevance can be computed even faster and, as the name says, approximates the optimized relevance.

7.1 Basic Relevance

The DECAY FACTOR of an article a at the instant of time t with respect to a decay half-time k is defined as

$$\omega_{k,t}(a) = \left(\frac{1}{2}\right)^{age_t(a)/k} .$$

Let a be an article, let u be a user, and let t be an instant of time. The BASIC RELEVANCE $\rho_t(a,u)$ of article a with respect to user u at the instant of time t is

$$\begin{aligned}\rho_t(a,u) &= (f_a \times g_{u,t})\omega_{k,t}(a) \\ &= \sum_{w \in \mathcal{W}} (f_a(w) g_{u,t}(w))\, \omega_{k,t}(a) .\end{aligned}$$

Notice that $0 \leq \rho_t(a,u) \leq 1$. Thus, when $\rho_t(a,u)$ is close to 1, article a is relevant to user u at the instant of time t. When instead $\rho_t(a,u)$ is close to 0, article a is not relevant to user u at the instant of time t.

Next, we analyze the time complexity of the basic relevance algorithm, that is, the time complexity needed for computing the basic relevances of all articles in the candidate set $\mathcal{C}_t$. This computation can be performed in two steps: first $g_{u,t}$ is computed, and then $\rho_t(a,u)$ is computed, for all candidate articles $a \in \mathcal{C}_t$.

Denote with μ the average number of words occurring in an article. Then, the first step takes average time $O(|liking_{u,t}|\mu)$. Given a candidate article a, computing $\rho_t(a,u)$ takes average time $O(\mu)$. Therefore, the second step takes average time $O(|\mathcal{C}_t|\mu)$. Therefore, the average case time complexity of the basic relevance algorithm is $O((|liking_{u,t}| + |\mathcal{C}_t|)\mu)$. Since in practice $|liking_{u,t}| < |\mathcal{C}_t|$, this complexity can be simplified to $O(|\mathcal{C}_t|\mu)$.

7.2 Optimized Relevance

Let a be an article, let u be a user, and let t be an instant of time. The OPTIMIZED RELEVANCE $\rho_t^\infty(a,u)$ of article a with respect to user u at the instant of time t is

$$\rho_t^\infty(a,u) = \left(\sum_{b \in liking_{u,t}} \sigma(a,b) \right) \omega_{k,t}(a) .$$

The optimized relevance is equal to the basic relevance, modulo a factor that depends only on u and t. More precisely, it is easy to verify that

$$\rho_t(a,u) = \beta \rho_t^\infty(a,u) ,$$

where $\beta = |g'_{u,t}|^{-1}$.

Next, we analyze the time complexity of the optimized relevance algorithm, that is, the time complexity needed for computing the optimized relevances of all articles in the candidate set $\mathcal{C}_t$. Optimized relevances can be computed efficiently online if the similarity $\sigma(a,b)$ between articles has already been computed offline. Indeed, given an article a, the computation of $\rho_t^\infty(a,u)$ can be done in time $O(|\,liking_{u,t}\,|)$. Therefore, the time complexity of the optimized relevance algorithm is $O(|\mathcal{C}_t||\,liking_{u,t}\,|)$.

In practice, $|\,liking_{u,t}\,| < \mu$, and therefore the optimized relevance algorithm is faster than the basic relevance algorithm.

7.3 Approximated Relevance

In order to compute approximated relevances, we fix a small positive constant m, so that $m << |\mathcal{C}_t|$. At each instant of time t, for each article $b \in \mathcal{A}_t$, we keep a set $\mathcal{N}_t^m(b)$ containing the m articles a in $\mathcal{A}_t$ with the highest value of $\sigma(a,b)\omega_{k,t}(a)$.

Let a be an article, let u be a user, and let t be an instant of time. The m-APPROXIMATED RELEVANCE $\rho_t^m(a,u)$ of article a with respect to user u at the instant of time t is

$$\rho_t^m(a,u) = \left(\sum_{\substack{b \in liking_{u,t} \\ a \in \mathcal{N}_t^m(b)}} \sigma(a,b) \right) \omega_{k,t}(a) .$$

Note that, as m tends to infinite, the m-approximated relevance tends to the optimized relevance, that is

$$\lim_{m\to\infty} \rho_t^m(a,u) = \rho_t^\infty(a,u) .$$

Next, we analyze the time complexity of the m-approximated relevance algorithm, that is, the time complexity needed for computing the m-approximated relevances of all articles in the candidate set $\mathcal{C}_t$.

Clearly, this complexity is equal to $O(m|\,liking_{u,t}\,|)$. If m is assumed to be a small constant, this complexity reduces to $O(|\,liking_{u,t}\,|)$.

It follows that the m-approximated relevance algorithm is much faster than both the optimized relevance algorithm and the basic relevance algorithm.

8 CONCLUSIONS

We have presented a recommendation algorithm for online news based on collective intelligence and content. When a user asks for personalized news, we recommend those articles that (i) are popular among the members of the online community (the collective intelligence part), and (ii) are similar in content to the articles the user has read in the past (the content part).

Our algorithm computes its recomendations based on the collective behavior of the online users, as well as on the feedback the users provide to the algorithm's recommendations. The users' feedback can moreover be used to measure the effectiveness of our recomendation algorithm in terms of the information retrieval concepts of precision and recall.

The cornerstone of our recommendation system is a basic relevance algorithm that computes how relevant an article is to a given user. The basic relevance algorithm gives very high quality results, but is not scalable because it takes time proportional to the product of the number of candidate articles with the average number of distinct words contained in an article.

To address the scalability problem, we devised an optimized relevance algorithm that gives the same results of the basic relevance algorithm at the cost of minimal offline computations. The optimized relevance algorithm takes time proportional to product of the number of candidate articles with the number of articles explicitly liked by the user. It is therefore faster than the basic relevance algorithm, but not yet scalable.

To obtain a scalable algorithm, we have developed an m-approximated relevance algorithm, where m is a fixed constant. The m-approximated relevance algorithm is scalable, and it takes time proportional to the number of articles explicitly liked by the user.

REFERENCES

Banos, E., Katakis, I., Bassiliades, N., Tsoumakas, G., and Vlahavas, I. P. (2006). PersoNews: A personalized news reader enhanced by machine learning and semantic filtering. In *Ontologies, DataBases, and Applications of Semantics*, pages 975–982.

Bharat, K., Kamba, T., and Albers, M. (1998). Personalized, interactive news on the web. *Multimedia Systems*, 6(5):349–358.

Borsje, J., Levering, L., and Frasincar, F. (2008). Hermes: A semantic web-based news decision support system. In *Symposium on Applied Computing*, pages 2415–2420.

Das, A., Datar, M., Garg, A., and Rajaram, S. (2007). Google news personalization: Scalable online collaborative filtering. In *International Conference on World Wide Web*, pages 271–280.

Lang, K. (1995). Newsweeder: Learning to filter netnews. In *International Conference on Machine Learning*, pages 331–339.

Pazzani, M. J. and Billsus, D. (2007). Content-based recommendation systems. In *The Adaptive Web*, pages 325–341.

Segaran, T. (2007). *Programming Collective Intelligence: Building Smart Web 2.0 Applications*. O'Reilly.