Collaborative Filtering inspired from Language Modeling

Geoffray Bonnin
geoffray.bonnin@loria.fr

Armelle Brun
armelle.brun@loria.fr

Anne Boyer
anne.boyer@loria.fr

LORIA – KIWI Team
615, Rue du Jardin Botanique
54600 Villers-Lès-Nancy, France

Abstract

Recommender systems filter resources for a given user by predicting the most pertinent item given a specific context. This paper describes a new approach of generating suitable recommendations based on the active user's navigation stream. The underlying hypothesis is that the items order in the stream results from the intrinsic logic of the user's behavior.

We show similarities between natural language and Internet navigation and put forward navigation specificities. We then design a new model that integrates advantages of statistical language models such as n-grams and triggers to compute recommendations.

The resulting Sequence Based Recommender has been tested on Internet navigation artificial corpora.

1 Introduction

Recommender Systems (RS) [9] filter resources (movies, music, web pages, etc.) for a given user by predicting the most pertinent resource given a specific context. Nowadays, these systems are becoming more and more important: they provide significant profits to e-commerce companies and represent a way to find pertinent resources among the huge amount of data available online, they facilitate the interaction of users in online communities, etc. One approach to compute recommendations is the Collaborative Filtering (CF).

CF [7, 5] collects opinions from users about items (for example, ratings). When asked for a recommendation, RS identifies similar users and suggests the items these users have liked in the past. The algorithm does not need any representation of the items in term of features. It is only based on the ratings of the users community. We argue that recommendations not only depend on the previous items rated, but also on the order in which these items are consulted. For example, given two often commonly liked items i and j, it may be possible that consulting item i before item j is more pertinent than consulting item j before item i. It would then be more accurate for a recommender system to first suggest item i and then to recommend item j. We thus propose to integrate the notion of sequences to the CF framework that usually ignores it.

To achieve this objective, we borrow some language modeling tools: indeed, both domains CF modeling and Language modeling are similar in several points [2]. As statistical approaches lead to efficient language modeling, we study how models based on statistical language models can improve personalized recommendations of resources, by taking into account sequentiality and integrating navigation specificities. We extract frequent patterns of consultation by taking advantage of the analogy between language-based statistical modeling and resource retrieval. The resulting model-based CF algorithm computes the pertinence of a resource for a given user in a sequential context, based on the hypothesis that there is logic and coherency defining implicit rules within a navigation.

The rest of this paper is organized as follows. In Section 2 the similarities between natural language and Internet navigation are put forward. Then, we address the general issue of applying statistical language models to make recommendations. Based on this discussion, Section 3 defines our sequence-based model and the variants we experimented. In Section 4 we present the way we evaluated the variants of our model and in Section 5 the results obtained. We conclude in section 6 and present perspectives in section 7.

2 Context and Motivation

2.1 Natural Language vs Internet Navigation

Web browsing and natural language are similar in several points:

- Language modeling uses a vocabulary made up of words which can be viewed as being similar to the set of resources R of the web.
- A sequence of S words in a sentence is similar to a sequence of consultation of S resources.
- The presence of a word in a sentence mainly depends on its previous words, as the consultation of a resource mainly depends of the preceding consultations.

Given these similarities, we can naturally investigate the exploitation of language modeling techniques into a CF system. However two main differences exist between natural language and Internet navigation: (1) it is possible to have several Internet navigations overlapped, which would correspond to mixed sentences in natural language that does not exist, (2) natural language is governed by strong constraints: each word and its localization in a sentence is important ; navigation is less constrained and should be processed with more permissive models.

Thus, the model must be able to handle parallel navigations, which means the considered history should be longer than those of language modeling: the items of one session can be relatively distant. The second difference is more fortunate as the large number of resources available on the Internet forces the use of rather light models, thus a more permissive configuration is welcome. On the other side, it implies that the models are able to consider a relatively long history too: indeed the two or three first resources back in a navigation history are not as informative as in natural language.

2.2 Applying Statistical Language Modeling

Two models in the field of Statistical Language Modeling are predominant: n-grams [1] and triggers [10].

n-gram models simply consider the $n-1$ previous words in the history. In practice, $n = 3$ or 4, rarely 5. Indeed, the possible combinations above these values are too huge to find enough training data able to fill them in a representative way, and even if possible, the obtained models are too heavy. Thus, such models are rather simple but do not capture a very long history.

Trigger models are able to capture long-term dependences between words. It is made up of highly correlated ordered-pairs of words, the first one being the trigger and the second the triggered word. The advantage of such a model is its robustness to noise, due to the window used for the computation of correlations and for the computation of the probability of a word given its history. However, such a model usually takes into account only relationships between two words, and only the most correlated pairs of words are integrated in the model.

Few articles deal with *multi-word triggers*, which consider more than two words as triggers. Actually the only one we found is [4], in which triggers consist in pairs of words. Here again, only the most correlated words are integrated.

Trigger models cannot be used alone as only the subset of the most highly correlated pairs is retained. Thus, they are often used to refine n-grams models [4, 10]. Such a configuration is appropriate in natural language: the closest words in history are the most informative to predict the next one. But as said in Section 2.1, the field of recommendation is less constrained. So the closest previous resources may not be informative enough to predict the next one. For these reasons we propose a model able to capture long-distance dependencies while allowing a larger coverage than common triggers. The model we found to fulfill these requirements is the skipping model.

Skipping [3, 6] is simply an enhancement of n-gram models, allowing non-contiguous n-grams. The usual way to use this principle is to build skipped k-gram submodels, $k < n$, and to combine them afterwards with the n-gram model. The way we propose to use it is a little different: it consists in adding occurrences of skipped n-grams within a window to the counts of raw n-grams (contiguous n-grams).

2.3 Sequentiality in Information Delivery

Few publications are interested in the prediction of a resource's pertinence, as a sequential problem.

In the frame of recommendations in digital libraries, [8] computes the probability of the next visited document (item) given the most recently visited document (bigrams) and past indicative documents (triggers). However, in this work ratings are not taken into account, they only consider that the documents have been seen or not. Moreover, these probabilities are computed inside a unique session.

In [11], a n-gram based model is used to initialize a MDP recommender system. This n-gram based model integrates skipping, assuming that the probability of the next item depends on items other than just the previous $n-1$. This model has been compared to a dependency network based model in which the local distributions are probabilistic decision trees. Although these algorithms are among the most competitive, the n-gram based model obtained better results.

3 Sequence Based Recommender

The SBR model we propose integrates skipping in n-grams models in order to be robust to noise and to to be able to handle parallel navigations. Several skipping variants are first studied. Then, the functioning of our sequence-based model is detailed.

3.1 Variants

Shani's

This variant comes from [11]. It consists in allowing skipping only for the last element of the n-grams, all other elements being contiguous.

For example: let $n = 3$, and (a, b, x, y, z, c, d) a navigation sequence where (a, b, c, d) and (x, y, z) correspond to overlapped navigations. Instead of considering only raw triplets as (a, b, x) or (y, z, c), this variant allows to consider triplets as (a, b, y) or (a, b, c) too. This variant is thus able to capture a navigation overlap if the last element corresponds to a new navigation as for the triplet (a, b, c) ; however, it is not the case if the navigation already began: for example, the triplet (b, c, d) cannot be handled as b and c are not contiguous.

Enhanced skipping

This variant was especially designed to take into account parallel navigations: as the previous variant is not able to detect all of the sequences from a same navigation, we propose to consider both configurations simultaneously: skipping the last and the first element of the n-gram.

Example: applied on the same example as for the previous variant, it becomes possible to handle both the case (a, b, c) and the case (b, c, d).

Full skipping

This variant we propose goes a step further by allowing skipping between all of the elements of the n-gram.

3.2 Process

Our model relies on the following steps:
step 1 Training the model on a corpus to determine the counts of the (raw and skipped) n-grams
step 2 Computing the probabilities of the n-grams based on their counts
step 3 Running the model to predict the pertinent resources to the active user given his/her stream of navigation

Training

Let $S = \{s_1 \dots s_L\}$ a set of sessions of navigation, $R = \{r_1 \dots r_M\}$ the set of resources with ratings *i.e.* r_i is a pair (item, rating). A session $s_i = r_1 \dots r_{|s_i|}$ is made up of a sequence of N rated resources.

In the training phase, raw n-grams and skipped n-grams are considered. The question is how to add occurrences of skipped n-grams to the occurrences of raw n-grams. Notice that the maximum distance to take into account in a skipped n-grams model has to be fixed. This distance is denoted as D.

The occurrences of the skipped n-grams are added to the counts of their corresponding raw n-grams. Algorithm 1 presents how these counts are computed when $n = 3$.

```
Data: a set S of navigation sessions
Result: a list of trigrams associated with their
        occurrences
trigramlist ← ();
for each session s = (r_1 ... r_|s|) in S do
    for i ← 1 to |s| − 2 do
        for j ← i + 1 to min(i + D, |s| − 1) do
            for k ← j + 1 to min(j + 1 + D, |s|) do
                trigram ← (r_i, r_j, r_k);
                if trigram is in trigramlist then
                    increment its occurrences
                else
                    add trigram to trigramlist with
                    occurrence = 1
                end
            end
        end
    end
end
```

Algorithm 1: Computing counts of trigrams (Step 1)

Computing probabilities of the n-grams

Given the filtered n-grams counts, the conditional probabilities have to be computed. Let the trigram (r_1, r_2, r_3), the probability of resource r_3 given resources r_1, r_2 is computed as follows:

$$P(r_3 \mid r_1, r_2) = \frac{C(r_1, r_2, r_3)}{C(r_1, r_2)} \tag{1}$$

where $C(r_1, r_2, r_3)$ is the count of the skipped trigram and $C(r_1, r_2)$ is the count of the skipped bigram.

Recommendations

Once the probabilities of the model have been computed, the next step is the computation of predictions. For each candidate item, its probability is computed according to its history and the item that will be predicted is the one that maximizes the following equation:

$$r_i^* = \underset{i}{argmax} \quad P(r_i|h) \tag{2}$$

The problem is reduced to the computation of the probability of a resource given its history. In [11], Shani *et al.* considered an history limited to the contiguous $n - 1$ preceding resources. However, as our model exploits distance

between all resources of the n-grams during training, we argue that performance may be increased if all distant resources are considered for prediction (in the limit of D).

If $n = 3$, $p(r_i \mid h)$ is computed according the probability of each skipped trigram: $(r_1, r_2, r_i), (r_1, r_{i-1}, r_i)$, etc.

The probability of resource r_i given its history is computed as follows:

$$P(r_i \mid h) = 1 - \left(\prod_{j=i-D}^{i-2} \prod_{k=j+1}^{i-1} (1 - (p(r_i \mid r_j r_k))) \right) \quad (3)$$

This formula takes into account all skipped trigram histories, computes the probability of r_i given each of these trigram histories.

In the results we present in Section 5, tests will be performed both using only the contiguous $n-1$ previous resources to compute predictions as used in [11] and using skipping for prediction. We argue that using skipping for prediction will increase performance ; however computation time will increase too. That's why computing predictions in a real-time framework leads to a limitation of the skipping distance.

The way to compute probabilities with the preceding formula has the advantage to be anytime: it allows to progressively incorporate a more and more distant history. for example, let two histories h_1 and h_2 with respective probabilities p_1 and p_2 to lead to resource r. Both history have different distance. Using these histories results in the following probability:

$$P(r_i \mid h) = 1 - (1 - p_1)(1 - p_2) \quad (4)$$

Given an even more distant third history h_3 with the probability p_3, the model would then set the new probability associated to r as:

$$\begin{aligned} P'(r_i \mid h) &= 1 - (1 - P(r_i \mid h))(1 - p_3) \\ &= 1 - (1 - p_1)(1 - p_2)(1 - p_3) \end{aligned} \quad (5)$$

Thus, in a real-time framework, the computation of $P(r_i \mid h)$ may be computed given the contiguous history, and if additional time is available, a longer history may be taken into account to compute $P(r_i \mid h)$.

4 Evaluation Protocol

4.1 Corpus

Our sequence-based model has been tested on artificial corpora. The reason why we chose such corpora are the following: (1) our model has been designed to be used in an Internet navigation context. As such corpora are difficult to obtain, we decided to build artificial ones. (2) using parameterizable artificial corpora allows us to study how the model and its variants behave according to these parameters and thus to determine which configuration makes it the most efficient and robust.

In order to make them as realistic as possible, several criterion have to be included into the corpora. First, they should contain both sequential and non sequential navigations. Second, they should allow navigation overlapping, as it is often the case in Internet navigation. For example, a user can begin to navigate on a webmail, then switch to the meteo, and finally return to its webmail. Third, noise must figure in it as Internet users often do mistakes while navigating. Fourth, it is possible that a user brutally interrupts a session, we thus introduced this possibility. Last, the notion of communities may be included: indeed, the type of Internet navigation highly relies on the type of user. We thus build homogeneous corpora, in which users have a common behaviour. When using a real corpus, we plan to first apply a user clustering in order to obtain such homogeneous corpora.

The artificial corpora are generated in the following way:

1. Create the set of resources R of size $|R|$.
2. Create a core consisting in sets of sequences and unordered sets of correlated resources. The size of both sequences and sets range from v_{min} to v_{max}. These sequences and sets, use $z\%$ of the available resources. Ratings from 1 to 5 are randomly assigned to the resources.
3. Randomly choose a number of sessions for each user.
4. For each user
   (a) Randomly choose more than $\sigma\%$ sequences and sets from the core.
   (b) Add other sets of sequences and unordered sets of correlated resources chosen in the same as in 2.
   (c) Build sessions by concatenating and overlapping the previously created sequences and sets. Overlapping corresponds to parallel navigations. At this step, two features are included: sequences can be swapped with probability p_s or interrupted with probability p_i. The number of parallel sessions (due to swapping) does not exceed μ. The minimum size of sessions (due to interruption) is set to S_{min} and the maximum size is set to S_{max}.
5. Noise is inserted randomly into the whole corpus using all available resources (including resources that are not part of sequences and sets).

6. The resulting corpus is made up of $|T|$ items and is divided into a training and a test set. $\lambda\%$ are dedicated for the training set and the remaining is used for the test.

The artificial corpora used in our experiments have been built according to the parameters presented in Table 1.

Table 1. Parameter values of the corpus generated

Parameter	value
$\|R\|$	1000
v_{min}	3
v_{max}	10
σ	50
z	90
p_s	0.20
p_i	0.05
μ	2
S_{lin}	3
S_{max}	20
$\|T\|$	$1,000,000$
λ	90

4.2 Evaluation Metric

To evaluate our models, we used the Top-m rate metric. This metric is meant to evaluate the average pertinence of recommendation lists. For each history of the test corpus, a recommendation list of size m is built, containing the m most probable resources having a high rating (i.e. $\geqslant 4$) according to the model. If the actual resource is in the list, the recommendation is pertinent. We call this measure the Top-m rate metric. Thus, as a resource is recommended only when its predicted rating is high, the Top-m metric does not consider the cases when the observed resource has a low rating (*i.e.* $\leqslant 3$) and is done only on the highly rated observed resources.

One particular case has to be taken into account: when making a recommendation, it is possible that a history of the test corpus is not found in the model (meaning it was not encountered during the training step). In such a case, nothing can be recommended, the Top-m recommendation list is empty, and obviously the actual resource is outside of it. We thus considered these cases as being non-pertinent.

5 Experimental Results

This section is dedicated to the study of the robustness to noise of the SBR model. In-depth tests are first presented on the artificial corpora in order to show which parameters and variants are the most efficient and robust to noise.

In Figure 1, the different skipping variants presented in Section 3 are applied in their trigram form. The skipping is used with a scope of 8, only during the training step, no skipping is used for prediction: only the two previous resources are used to make the recommendation lists, as made in [11]. As can be seen, all the models provide almost the same accuracy. We can conclude that the weighting scheme does not lead to different performance when used only at training.

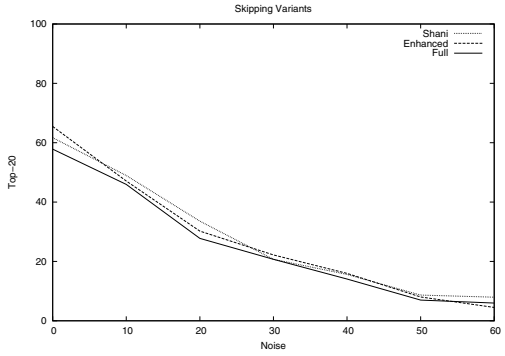

Figure 1. Top-20 rate when using skipping only during training.

We are now interested in the study of the impact of using distant information (distant n-grams) during the prediction step. We show the results we get when simply applying skipping variants with a scope of 8.

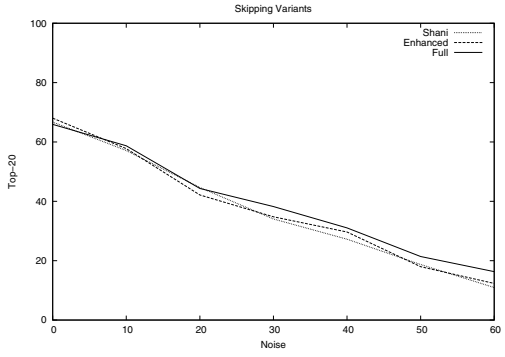

Figure 2. Top-20 rate when using skipping during both training and prediction.

First, we can see that the three models have been improved. Once again they provide almost the same results. Nevertheless, when looking at the proportion of covered

Noise	0	10	20	30	40	50	60
Shani	78%	70%	61%	53%	46%	39%	30%
Enhanced	83%	74%	67%	61%	56%	49%	42%
Full	83%	76%	68%	64%	59%	54%	51%

Table 2. Coverage when using skipping during both training and prediction

cases, we can see that the full skipping model is able to make more recommendations without reducing their quality, actually it even provides slightly better results. This can be explained by the fact that all the elements of the trigrams can be skipped, leading to a model that considers a higher number of trigrams. However its complexity is higher than the one of the two other variants. For a scope of 8, its complexity is 8 times larger than the one of the Shani skipping variant ; the size of the obtained model is larger too, which increases again the runtime. On our data sets, the full skipping variant provided models about twice as big as the enhanced ones, which are themselves about twice as big as the ones of the Shani skipping variant. Moreover, the enhanced variant has a complexity only twice higher than the one of the Shani variant and provides both better results and coverage. Thus, it can be an interesting alternative to the full skipping variant depending on the environment.

6 Conclusion

In this article, we argue that users ratings not only depend on the previous items rated, but also on the order in which these items have been consulted. We thus draw a parallel between natural language and Internet navigation and conclude that we can take inspiration of statistical language models to perform predictions in the frame of navigation. We propose a new model that takes advantage of two well performing statistical language models: n-grams and triggers. This model, called SBR (Sequence Based Recommender) is based on a n-grams model and is robust to noise as it integrates skipping to use resources consulted before the $n-1$ previous resources. It has the advantage to take into account long histories during prediction while being anytime: it can use only the previous $n-1$ resources and can also exploit a longer history, if time is available. Last, this model is a tractable model as its size is equivalent to a trigram model while considering long histories.

Our model has been tested on artificial corpora in order to determine which properties are the most appropriate to noise and parallel navigation. Two variants came up, the first one providing the best accuracy and coverage, and the second one being not quite as good but having a better runtime performance.

7 Perspectives

Shortly, we intend to test our algorithm on real data sets: news, music, movies (Netflix) and web navigation real data sets.

In a future work, we will study the way to integrate orderless sequences (sets of sequences) as the ones inserted in the corpus by the generator algorithm. A way to achieve this objective is to merge n-grams: all the permutations of a n-grams are merged if they all have a similar frequency.

Another way of improvement if to compute classes of items, when their conditional probabilities are shown equivalent. Using classes of items also allows to cope with sparse data problem, as n-classes models do in statistical language modeling.

Last we plan to combine our sequence based model with a state of the art one by first applying the state of the art model and then refine the items to recommend according to the user history.

References

[1] L. R. Bahl, F. Jelinek, and R. L. Mercer. "a maximum likelihood approach to continuous speech recognition". *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 5(2):179–190, 1983.

[2] A. Boyer and A. Brun. Usage Based Indexing of Web Resources with Natural Language Processing. 2007.

[3] S. F. Chen and J. T. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 1999.

[4] Y. Chen and K.-P. Chan. Extended multi-word trigger pair language model using data mining technique. *IEEE International Conference on Systems, Man and Cybernetics*, 1:262–267, 2003.

[5] A. Das, M. Datar, A. Garg, and S. Rajaram. Google News Personalization: Scalable Online Collaborative Filtering. *WWW 2007 Conference*, 2007.

[6] J. Gao, H. Suzuki, and Y. Suzuki, H. Wen. Exploiting headword dependency and predictive clustering for language modeling. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 248 – 256, 2002.

[7] D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[8] D. Pavlov, A. Manavoglu, and C. Giles. Collaborative filtering with maximum entropy. *IEEE Intelligent Systems*, 2004.

[9] P. Resnick and H. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[10] R. Rosenfeld and X. Huang. Improvement in stochastic language modeling. In *DARPA Workshop on Speech and Natural Language*, pages 107–111, San Mateo,CA, 1992.

[11] G. Shani, D. Heckerman, and R. I. Brafman. An MDP-Based Recommender System. *The Journal of Machine Learning Research*, 2005.