A Top-N recommender model with partially predefined structure

El Mehdi Rochd*† and Mohamed Quafafou*
*Aix-Marseille University, LSIS UMR 7296, Marseille, France
†Marketshot, Paris, France
{el-mehdi.rochd,mohamed.quafafou}@univ-amu.fr

***Abstract*—Recommender systems can retrieve appropriate results based on users behavioral patterns and preferences. They may be built based on multi-label learning approaches, as each customer transaction may be labeled with several results that interest him/her. It is therefore useful to model the correlations between labels while controlling complexity of the learning algorithm. This paper presents a generative probabilistic model for online resources (products/URLs) recommendation, by capturing the complex local correspondence between the user's queries and the resources he/she has actually viewed. The structure of our model is partially defined and it is completed according to the observed data. Consequently, several links between observed and/or latent random variables are induced from the training dataset before starting the estimation of parameters. Experiments conducted on real data show the effectiveness of our approach.**

***Keywords*-Topic Models, Recommendation, User Behavior, E-commerce, Local Influence, Information Retrieval.**

I. Introduction

The webspace is becoming an attractive business plateform that can help limiting the number of middlemen between product/service providers and their customers and thereby increasing direct income. Unfortunately, a lot of competition makes the market offer more diverse but at the same time overwhelming. Consequently, how to enable Internet users to avoid the impression that they are drowning in information? Recommender systems can retrieve the appropriate results based on past users behavior patterns and preferences. They may be built based on multi-label learning approaches, as each customer transaction is labeled with several results interesting for him/her. Then, it follows the construction of effective models for detecting the interests of users [11] and generating user profiles that are exploited by some search engines in order to make recommendations. However, search engines rank results based primarily on the submitted user queries. Nevertheless, the same query could be used in different contexts since individual users have different interests. To improve the relevance of search results, it is necessary to re-sort the ranked lists according to the learned user profile. Thus, by having some information about the user issuing a query and a prior knowledge of his/her previous search behavior, we can adapt the ranked lists so that the likelihood of highly rated results being relevant is increased.

In this context, traditional methods such as web usage mining techniques [18] allow to discover new trends in the user's browsing behavior from log files. Identifying these trends can categorize the types of browsing behavior. Other approaches, such as sessions clustering [5] or association rules [19] are also used to classify users according to the similarity of their behavior. These methods do not generally provide the ability to automatically characterize the unobservable factors that lead to common navigational patterns. To overcome this limitation, we extend the probabilistic topic models [2] for the task of recommendation. In fact, topic models have demonstrated effectiveness in the discovery of the underlying topics in text documents by learning the groups of semantically consistent words that generate the training data. Thereby, we can get a representation of the topical interests of a user taking into account the searches carried out by previous users, the query items used in these searches and the results which were selected or clicked by users.

In this work, we use query logs from two real-world datasets based on real users. The first one is composed of queries from a website that compares online products based on users'search criteria. The second one is the AOL Search dataset, which is a collection of real query log data. We consider that the user profiles are constructed from the representation of the results (products or/and URLs) that were selected by the users over a topic space. We use latent topic models to determine these topics by assuming that a topic is a probability distribution over the user's query (composed of words). Our contribution is to decompose the user session into a pair (query,result) observed from data, then to analyze the correspondence between the query and the result after automatically determining the local correlations between the results of each user session. In fact, in the multi-label learning research field, the well-known label-independence assumption makes easy the learning process by transforming a multi-label problem into one or more single label problems. A crucial problem with this approach is that it is unable to take advantage of labels correlation. It is therefore useful to model such correlation while controlling complexity of the learning algorithm.

Our experiments show that by locally exploiting correlations between results as well as the correspondence between queries and results, we can provide lists, the classification of which is significantly improved.

II. Related Work

In this section, we present some of the research literature related to recommender systems.

There are two popular techniques used for recommendation: content-based filtering systems and collaborative filtering systems. The filtering systems based on the content generate recommendations based on the pre-built user profiles by measuring the similarity of web content in these profiles. In contrast, collaborative filtering systems make recommendations using the appetite rate of the current user for a product, based on the preferences of other similar users.

In the context of recommender systems based on the content, Letizia [13] is an implementation that extends the web-browser to track the browsing behavior of the user. It builds a custom model which consists of keywords related to the user's interests. It is based on implicit feedback to infer the user's preferences. In the context of collaborative filtering systems, Tapestry [7] is a system based on the explicit opinions of people in a close-knit community. However, when the size of the community becomes important, the recommendation system can not depend on each person knowing the other.

Other techniques have been applied for recommendation, including clustering and Bayesian networks. Clustering techniques [5] identify groups of users with similar preferences. After obtaining the groups, we can make predictions for a user by averaging the positions of other users belonging to this cluster. The advantage of this method is that it makes easier the analysis of large volumes of data, because the identified groups of users help to reduce the data to be processed. Bayesian networks [15] establish a model based on a training set with a decision tree at each node, where the edges represent the user information. We obtain a fast and small model, the accuracy of which is equivalent to that of the nearest neighbor methods [8]. Bayesian networks are useful in cases where information about the user's preferences does not vary hugely with respect to the time required to build the model, but are not suitable for environments where the user's preferences must be updated frequently. In [17], the authors present examples of recommendation systems used in e-commerce and how they can capture customer loyalty. Although these systems have been successful, they are exposed to some limitations such as the problems of sparsity in dataset [14] and problems in relation with the high dimensionality [12].

In the topic model community, many algorithms were developed for the task of recommendation. In [9], the authors proposed a framework for the discovery and analysis of web navigational patterns based on PLSA. This framework allows to characterize relationships between users and Web objects. Since these relationships are measured in terms of probabilities, probabilistic inference can be used to perform recommendation. Then, a collaborative web recommendation framework based on LDA was proposed in [20], which allows to discover the associations between web user sessions and topics. This framework outperforms the previous one.

The topic models cited above have a static structure and they are operating in the same way regardless of the data to be processed. To override this limit, a supervised model has been proposed, called loclda [16] whose structure is dynamic and adapts to any type of data. This model works by exploiting the local correlation between annotations. However, a threshold must be specified first in order to define the relationship between annotations, which may limit the effectiveness of the model.

In this paper, we propose to extend the LocLDA model with the aim of determining the relationship between annotations only from data architecture. Experiments conducted on real data demonstrate performance improvements.

III. The LocLDA model

Before introducing our approach, we present the web usage data model, which is the result of the data preprocessing phase. In fact, we have transformed the raw web log data into transaction data that we will then process to fulfill the task of recommendation. We obtain a set of S web user sessions $\{s_1, s_2, ..., s_S\}$. These sessions have led to the selection of N resources $\{w_1, w_2, ..., w_N\}$. A resource is the result selected by a user, it can be a product or a URL. The correlation between resources in a user session generates a neighborhood of resources including P parents of the user's session annotations. Finally, we have a set of M query items $\{q_1, q_2, ..., q_M\}$.

In the CorrLDA model, the correspondence between an annotation q_i and its associated resource is obtained via a latent variable y_i. We consider that y_i is the parent of q_i, we note: $y_i = \text{parent}(q_i)$. In LocLDA, we no longer consider that an annotation $q_i \in \{q_1, q_2, ..., q_M\}$ is connected to the resource $\mathcal{S}$ via a single latent variable, but through a set of latent variables y_j which are parents of $q_j \in \{q_1, q_2, ..., q_M\} - \{q_i\}$ annotations. Thus, we do not just consider the selected resources, but also the relationship between them, and the correspondence between these resources and user queries.

The advantage of this approach compared to that presented in [16] is that we no longer have to perform a regression on annotations to obtain their parents, but they are observed from the data.

A. The model description

In this section, we present our extension of the LocLDA, the graphical model of which is given in Figure 1.

Let $z = \{z_1, z_2, ..., z_M\}$ be the latent factors that generate the user session, and $y = \{y_1, y_2, ..., y_N\}$ be discrete

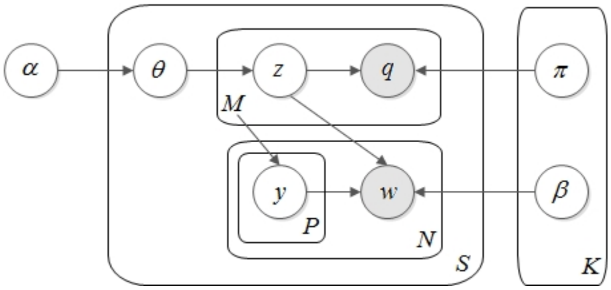

Figure 1. Graphical model of LocLDA

indexing variables that take values from *1* to *N* with equal probability. Conditioned on *N* and *M*, a *K*-factor LocLDA model assumes the following generative process for a pair session/resource (q, w):

1) Find the parents of each resource (from data).
2) Sample $\theta \sim \text{Dirichlet}(\theta|\alpha)$
3) For each query item $q_m, m \in \{1, ..., M\}$
   - Sample $z_m \sim \text{Multinomial}(\theta)$
   - Sample $q_m \sim p(q|z_m, \pi)$ from a multinomial distribution conditioned on z_m
4) For each resource $w_n, n \in \{1, ..., N\}$
   - Sample $y_n \sim \text{Uniform}(1, ..., N)$
   - Sample $w_n \sim p(w|y_n, y_p, \boldsymbol{z}, \beta)$

LocLDA defines the joint distribution of the query items, resources and latent variables as follows:

$$p(\boldsymbol{q}, \boldsymbol{w}, \theta, \boldsymbol{z}, \boldsymbol{y}|\alpha, \pi, \beta) = p(\theta|\alpha)\left(\prod_{m=1}^{M} p(z_m|\theta)p(q_m|z_m, \pi)\right)$$
$$\left(\prod_{n=1}^{N}\prod_{p} p(y_n|M)p(y_p|M)p(w_n|y_n, y_p, z, \beta)\right) \quad (1)$$

where α, π and β are the parameters we want to estimate.

B. Approximate Inference and Parameter Estimation

Since exact probabilistic inference is intractable for LocLDA, variational inference methods [10] are used to approximate the posterior distribution of the latent variables given a session/resource. Thus, a variational distribution h on the latent variables is introduced as follows:

$$h(\theta, \mathbf{z}, \mathbf{y}) = h(\theta|\gamma)\left(\prod_{m=1}^{M} h(z_m|\phi_m)\right)\left(\prod_{n=1}^{N} h(y_n|\lambda_n)\prod_{p} h(y_p|\lambda_p)\right) \quad (2)$$

where γ, ϕ and λ are variational parameters. Each variational parameter is appropriate to its respective random variable.

Then, we optimize the values of the variational parameters so that the variational distribution h is close to the true posterior p, by minimizing the Kullback-Leibler (KL) divergence between the variational distribution and the true posterior. In [16], it was shown that minimizing the KL divergence between the variational posterior probability and the true posterior probability is equivalent to maximizing the lower bound $L(\gamma, \phi, \lambda; \alpha, \pi, \beta)$, the expression of which is:

$$\begin{aligned} L(\gamma, \phi, \lambda; \alpha, \pi, \beta) \\ &= E_h[\log p(\theta|\alpha)] + E_h[\log p(\boldsymbol{z}|\theta)] + E_h[\log p(\boldsymbol{q}|\boldsymbol{z}, \pi)] \\ &+ E_h[\log p(\boldsymbol{y}|M)] + E_h[\log p(\boldsymbol{w}|\boldsymbol{y} \in \text{parents}(\boldsymbol{w}), z, \beta)] \\ &- E_h[\log h(\theta|\gamma)] - E_h[\log h(\boldsymbol{z}|\phi)] - E_h[\log h(\boldsymbol{y}|\lambda)] \quad (3) \end{aligned}$$

By expanding each term in equation (3) and setting to 0 the derivatives of L with respect to γ, ϕ and λ respectively, we get the following coordinate ascent algorithm:

1) Update the posterior Dirichlet parameters

$$\gamma_i = \alpha_i + \sum_{m=1}^{M} \phi_{mi} \quad (4)$$

2) For each query item, update the posterior distribution over topics

$$\phi_{mi} \propto \pi_{iq_m} \exp\left(\psi(\gamma_i) - \psi(\sum_{j=1}^{K} \gamma_j) + \sum_{n=1}^{N}\sum_{p} \lambda_{nm}\lambda_{pm} \log \beta_{iw_n}\right) \quad (5)$$

3) For each resource, update the posterior distribution over users'sessions

$$\lambda_{nm} \propto \exp\left(\sum_{i=1}^{K}\sum_{p} \phi_{mi}\lambda_{pm} \log \beta_{iw_n}\right) \quad (6)$$

These equations are invoked repeatedly until convergence.

The next step consists of maximizing the lower bound with respect to the model parameters α, π, β. Given a training set $S = \{(q_s, w_s)\}_{s=1}^{S}$, the objective is to find the maximum likelihood estimation for α, π, β. The corpus log-likelihood is bounded by :

$$L(S) = \sum_{s=1}^{S} \log p(q_s, w_s|\alpha, \pi, \beta) \geq \sum_{s=1}^{S} L(\gamma_s, \phi_s, \lambda_s; \alpha, \pi, \beta)$$

By considering terms containing π and β in L and setting to 0 the derivatives of L with respect to π and β (respectively), we get:

$$\pi_{ij} \propto \sum_{s=1}^{S}\sum_{m=1}^{M_s} \phi_{smi} q_{sm}^{j} \quad (7)$$

$$\beta_{ij} \propto \sum_{s=1}^{S}\sum_{n=1}^{N_s} w_{sn}^{j} \sum_{m=1}^{M_s}\sum_{p} \phi_{smi}\lambda_{snm}\lambda_{spm} \quad (8)$$

Finally, the Newton-Raphson algorithm [3] is used to estimate the Dirichlet α.

For further details on the calculations, see [16].

C. The Variational EM Algorithm

Once we have the parents of annotations in hand, we use a variational EM algorithm which performs iterative maximization of a lower bound of data in which some variables are unobserved. It maximizes a lower bound of the data log-likelihood with respect to the variational parameters, and then, for fixed values of the variational parameters, maximizes the lower bound with respect to the model parameters. Indeed, we have the following iterative algorithm:

- (E-Step) For each session, find the optimizing values of the variational parameters using equations (4), (5) and (6) with appropriate starting points for γ, ϕ_{mi} and λ_{nm}.
- (M-Step) Maximize the resulting lower bound on the log-likelihood with respect to the model parameters for fixed values of the variational parameters, using equations (7), (8) and the Newton-Raphson algorithm.

These two steps are repeated until the lower bound on the log-likelihood converges.

D. System architecture overview

Figure 2 shows the overview of our system's architecture. The extension we propose compared to [16] consists of defining the parents of a resource from the data to be processed rather than performing a regression on the resources with the constraint of setting thresholds to get the neighborhood of a resource.

IV. Ranking resources

In this section, we describe formulas for ranking resources using the parameters that were estimated based on the three models LocLDA, CorrLDA and LDA. The ojective is to return to the user a ranked set of resources ($w \in \mathcal{W}$) according to their likelihood given the user's query $Q = \{q_1, q_2, ..., q_n\}$ under each of the three models. The ranking in the case of LDA is:

$$\begin{aligned} p(w|Q) \propto p(w)p(Q|w) &= p(w) \prod_{q \in Q} p(q|w) \\ &= p(w) \prod_{q \in Q} \sum_{z} p(q|z)p(z|w) \quad (9) \end{aligned}$$

where:

$$p(w) = \frac{N_w}{N}$$

N_w is the number of words composing the user's query, which led to a click on the resource w.
N is the total number of words composing all user queries.

The ranking formula consists in multiplying a prior on the probability of the resource (which we denote $p(w)$) with the probability of the query given the resource (which we denote $p(Q|w)$). This latter quantity can be estimated by introducing latent topics from topic models. Indeed, topic models allow to estimate the probability of query items given topics $p(q|z)$ and the probability of topics given resources $p(z|w)$.

In the case of CorrLDA and LocLDA, once we have a trained model in hand, we can perform the variational inference (E-step) with fixed model parameters on a new user session to estimate the posterior of variational parameters. Furthermore, we can compute the conditional distributions of interest $p(w|S^*)$ where S^* is a user session which does not belong to the training set. The ranking formula which is a distribution over resources conditioned on new user session is approximated by:

$$p(w|S^*) = \sum_{z_m} p(z_m|\theta)p(w|z_m, \beta) \qquad (10)$$

V. Experiments

In this section, we describe the experiments we have conducted for recommending resources, in which we compare performances of LocLDA with those of CorrLDA and LDA.

A. Datasets

1) Marketshot dataset: Marketshot[1] is a company that generates qualified leads on the Internet through its websites. The dataset used in this paper is from the query logs of one of its website that connects potential buyers with major brands and distribution networks in the market of mobile telephony[2]. This website provides clear information about available products in the market of mobile telephony. It also provides various search and comparison tools in order to help the undecided users to choose the package that interests them the most, among the multitude of available products. We used a dataset based on a 1-month web log file (March, 2013). We generate the training data automatically from log file without any human intervention. To clean the data, we have kept the queries which have resulted in a product selection. Then, we have selected only the user sessions with more than 3 queries. This preprocessing stage is carried out to ensure that users have made a significant number of queries and that products were also selected reasonably, to ensure that user sessions are sufficiently annotated. Table I details the obtained corpus.

Our log file is composed mainly of seven attributes: the ID of the transaction, the ID of the user session, the mobile provider, the package, the package features, the user's query and the date when the user has made his query. Table II shows an example of one transaction from this query log. In our experiments, we consider that a product is represented by the triplet: (Package, Mobile Provider, Package features).

[1] http://www.marketshot.fr

[2] http://www.choisirsonforfait.com/

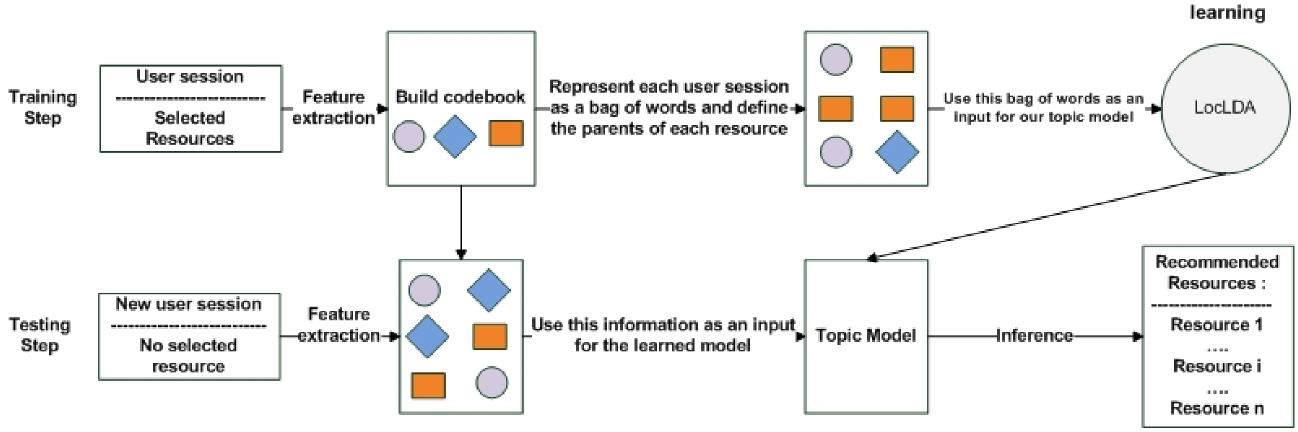

Figure 2. System architecture overview

Marketshot Dataset		AOL Dataset	
# Queries	21,300	# Queries	428,936
Training subset size	8,182	Training subset size	21,763
Testing subset size	413	Testing subset size	981
# Users	2,291	# Users	1,190
# Annotations	173	# Annotations	237
# Query items	199	# Query items	22,162

Table I
Counts for the datasets used in experimentation.

Id	**Id session**	**Package**	**Mobile Provider**	**Package features**	**date**	**user's request**
3	73f08ee8	Mobile plan 1	Mobile Provider *A*	2-years contract	2013-01-15 11:57:22	1 hour of calls, cell phone

Table II
Log file format used in our experiments.

2) AOL dataset: This publicly available query log dataset [4] is provided to the research community by AOL search engine[3]. It consists of ten files containing nearly 37M lines of data representing 657k users. We focus on the search events happened on May, 2006. Each search event is represented by a tuple (u, q, t), which means user u issued query q at time t, and we sort all the search events by their time. Then, we normalize queries through punctiation-removal and case-folding. User privacy was protected by analysing results only over aggregate data. The same preprocessing stage as above was performed for this dataset. In fact, since this dataset is huge, we considered arbitrarily one out of the ten available files and we selected the queries which resulted in a click on a URL. Then, the URLs for which more than 100 users had clicked on at least once were selected. Finally, we selected only those users with more than 100 remaining queries. The resulting reduced dataset is described in more detail in Table I.

B. Methodology

The cleaned data is separated in two subsets: training subset (the first 28 days of each dataset) and testing subset (the last day of each dataset). We have selected the last queries of each dataset for testing, to respect the order in which the queries were made. This means that the training and testing subsets follow the same chronological order. The objective is to learn profiles for 28 days in order to make recommendations for new sessions the day after. Concerning the parameter setting, we set the Dirichlet prior α to be $0.1/K$, where K is the number of topics used for experiments. We calculate the scores defined above for each query in the test set and rank the resources accoding to the values of these scores. We consider that a ranked resource is relevant if it is the same one the user had actually viewed. We evaluate the rankings by calculating the precision@n: p@10, p@15, p@20 by varying the number of topics. We stopped at rank 20, since in information retrieval, it is valuable that pertinent resources appear early in the ranked

[3] http://search.aol.com

list.

Then, we calculate the perplexity for the test sets of each dataset to assess the optimal number of topics for each model and each data. Indeed, in the experiments that follow, we will run the models using the optimal number of topics to be sure of the best possible quality of prediction. In order to determine if our model is improving the recommendation performance, we report another metric that we call the gain. This metric compares the number of times the LocLDA improves the ranking (which we denote $\#better$) to the number of times it worsens it (which we denote $\#worse$). A simple expression of this equation is given by:

$$\text{Gain} = \frac{\#better - \#worse}{\#better + \#worse}$$

When the value of this metric is 0, this means that there is no change between the LocLDA and the other models, when it is positive, then the LocLDA improves the ranking and when it is negative, the ranking is deteriorated.

VI. Results

A. Top-K resources-based evaluation

In this section, we focus on the relevance of the resources we recommend to users. Indeed, after the training step over a period covering 28 days for each dataset, we propose to automatically recommend resources to new users who have not selected any resource yet, by calculating the scores defined above. To assess the predictive power of the three models, we evaluate the precision of the ranked lists by varying the number of topics and the number of ranked resources (10, 15 and 20). Figures 3 and 4 show the results for the two datasets. As we can see, our approach enables to recommend resources with a precision exceeding that of CorrLDA and LDA, especially when 20 resources are recommended.

B. Perplexity

We compute the perplexity of the given annotations under $p(w|\boldsymbol{q})$ for each user session in the test set to measure the annotation quality of the models. In language modeling community, the perplexity is algebraically equivalent to the inverse of the geometric mean per-word likelihood [1], it is defined as:

$$\text{perplexity} = \exp\left(-\sum_{s=1}^{S}\sum_{n=1}^{N_s}\frac{\log p(w_m|\boldsymbol{q}_s)}{\sum_{s=1}^{S} N_s}\right) \tag{11}$$

Figure 5 illustrates the perplexity of the held-out annotations under the maximum likelihood estimates of each model for the two datasets (lower numbers are better). We observe that the lower values are obtained using LocLDA, we conclude that LocLDA finds much better predictive distributions of resources than CorrLDA and LDA. Thus, we will use in the next experiments 40 topics when running LocLDA and LDA, and 70 topics when running CorrLDA.

C. The effect of the query difficulty

There are number of measures of query difficulty [6]. In this paper, we will explore one measure, which is the click entropy for a query (when click data is available).

When a given session/query pair had been observed before, we can use this information about prior clicks by assuming that the user will again click on the same resources as before. However, in almost cases, the session/query pair will be novel and we will not have such prior information to exploit. When the query has been observed many times before, but always by other users, we are still able to use this information to provide a good ranking. In this section, we will introduce a measure called the *click entropy* to identify such unambiguous queries. The click entropy of an observed query q is defined as follows:

$$H_q = \sum_{w \in W(q)} -p(w|q)\log_2 p(w|q)$$

where $W(q)$ is the set of the selected resources given the query q and $p(w|q)$ is the frequency with which resource w was clicked amongst all the clicked resources given the query q. The entropy values vary in the range zero to the logarithm of the number of distinct resources clicked on for a query, consequently, the range of values depends on the query. This makes the comparison of click entropy values accross queries complicated. To deal with this issue, we will use, in our experiments, normalized entropy values instead, where the range of values is limited to $[0, 1]$, this new measure is defined as follows:

$$\widehat{H}_q = \frac{H_q}{\log_2 |W(q)|}$$

We have calculated this measure for all queries. We have separated these queries into two groups: queries for which this measure is lower than 0.5 and queries for which this measure is greater than 0.5. Then we calculated the Gain for each of the two groups that contain the test queries.
Figure 6 shows how the performance of the LocLDA changes as the normalized click entropy of the queries evolves.

We notice that the Gain increases as the click entropy decreases. In fact, the Gain for the LocLDA can reach up to 17% for queries, the normalized click entropy of which is lower than 0.5 compared to CorrLDA and it reaches 40% for queries, the normalized click entropy of which is lower than 0.5 compared to LDA.

VII. Conclusion and future work

In this paper, we have proposed an extension of the LocLDA model that recommend resources to users according to their search criteria. The contribution of this work is the decomposition of user session into a pair (query/resource) and exploit both the local correlations between resources selected by a user and the correspondence between the

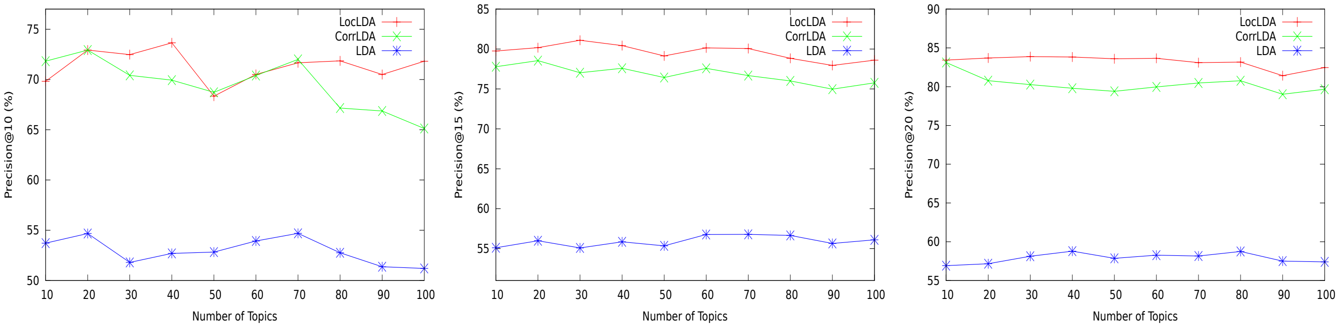

Figure 3. Ranking precision of the 3 approaches on the Marketshot dataset over all the test sessions.

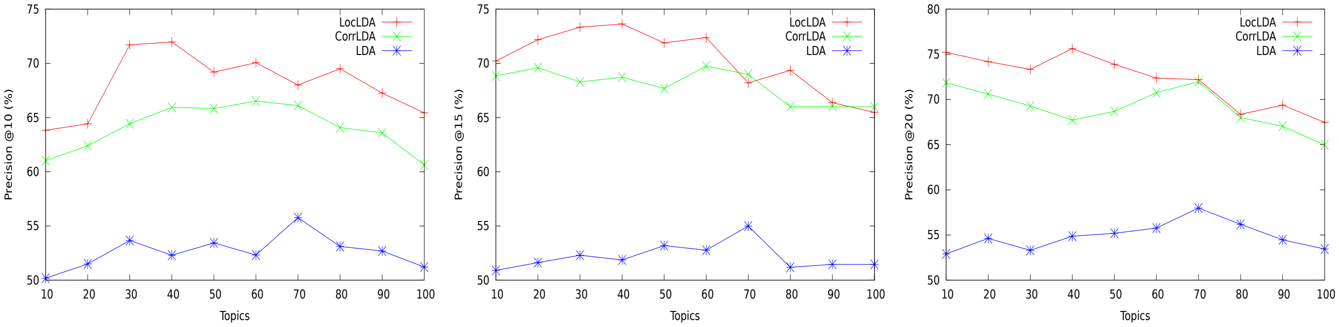

Figure 4. Ranking precision of the 3 approaches on the AOL dataset over all the test sessions.

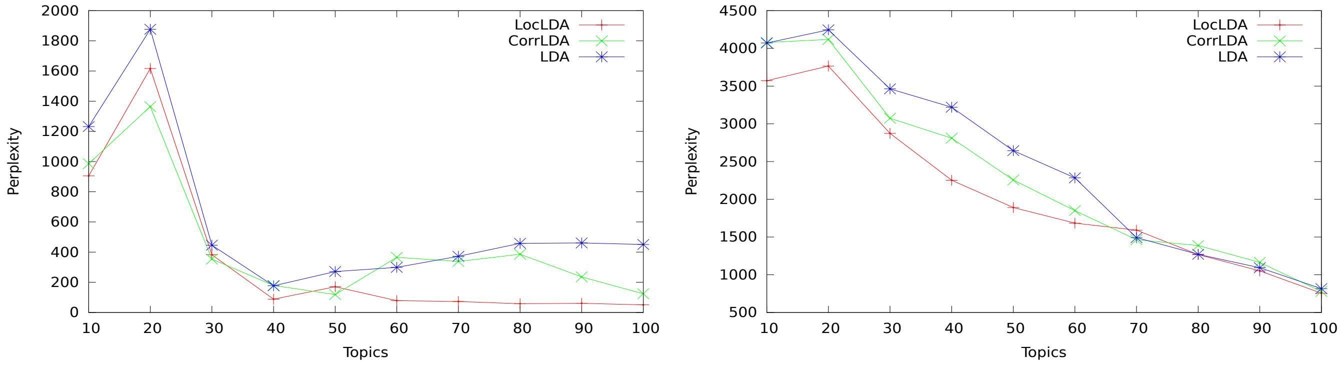

Figure 5. Perplexity on the test set for the ML estimates of the three models (lower numbers are better): (Left) Results for the Marketshot dataset, (Right) Results for the AOL dataset.

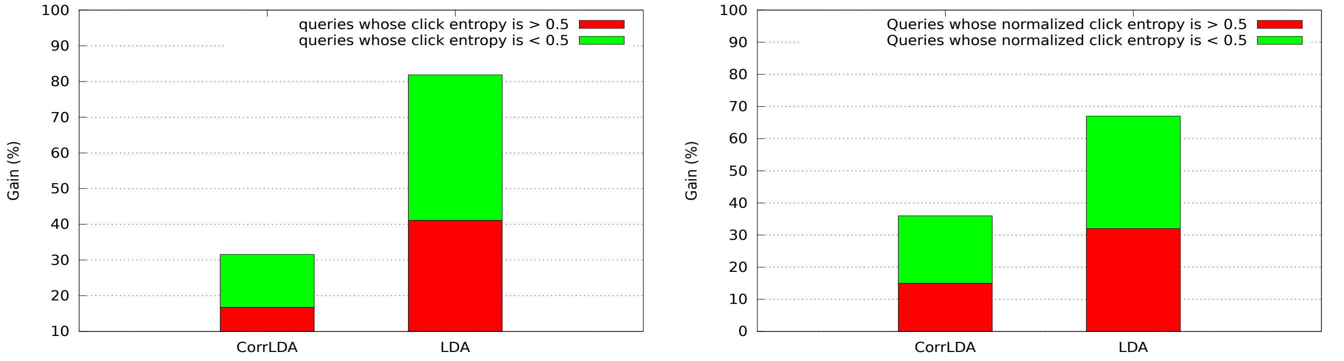

Figure 6. The effect of query ambiguity on the recommendation algorithms: (Left) Results for the Marketshot dataset, (Right) Results for the AOL dataset.

query items and the resources. We used two real-world datasets covering different periods in order to evaluate the recommendation performance of our model. The evaluation is done in terms of efficiency ranking. The results obtained using our model outperform those obtained by LDA and CorrLDA. In fact, using LocLDA, the returned resources in the top of the lists coincide with the interests of users, which is shown through the precision@n, that exceeds 80% when considering the top-20 resources. We also evaluated performances of our model using a query difficulty metric, which is the click entropy. Again, our model performs well and the obtained gain values are satisfactory. In our future work, we plan to incorporate explicitly the user profile in the model. In addition, we intend to introduce time dynamics under Markovian and non-Markovian fashions.

References

[1] D. Blei and M. Jordan. Modeling annotated data. *In ACM Conference on Research and Development in Informaion Retrieval*, SIGIR, 2003.

[2] D. M. Blei. Probabilistic topic models. *Commun. ACM, 55(4):77â84*, Apr. 2012.

[3] D. M. Blei, A. Y. Ng, M. I. Jordan, and J. Lafferty. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993â1022, 2003.

[4] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. *In Proceedings of the 1st International Conference on Scalable Information Systems, Infoscale*, 2006.

[5] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Model-based clustering and visualization of navigation patterns on a web site. *In Journal of Data Mining and Knowledge Discovery*, 7 (4), 2003.

[6] D. Carmel, E. Yom-Tov, A. Darlow, and D. Pelleg. What makes a query difficult? *In Proceedings of the international ACM conference on Research and development in Information Retrieval*, SIGIR, 2006.

[7] D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *In Communications of the ACM - Special issue on information filtering*. Volume 35 Issue 12, Dec. 1992.

[8] C. Holmes and N. Adams. A probabilistic nearest neighbour method for statistical pattern recognition. *In Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, Vol. 64, No. 2, 2002.

[9] X. Jin, Y. Zhou, and B. Mobasher. Web usage mining based on probabilistic latent semantic analysis. *In ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2004.

[10] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. *In Machine Learning*, 1999.

[11] J. Kim, K. Thompson, P. Bennett, and S. Dumais. Characterizing web content, user interests, and search behavior by reading level and topic. *In The International Conference on Web Search and Data Mining*, WSDM, 2012.

[12] H. Kriegel, P. Kroger, and A. Zimek. Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *In ACM Transactions on Knowledge Discovery from Data (TKDD)*, Volume 3 Issue 1, 2009.

[13] H. Lieberman. Letizia: an agent that assists web browsing. *In Proceedings of the 14th international joint conference on Artificial intelligence*, 1995.

[14] B. Piccart, J. Struyf, and H. Blockeel. Alleviating the sparsity problem in collaborative filtering by using an adapted distance and a graph-based method. *In Proceedings of the SIAM International Conference on Data Mining*, SDM, 2010.

[15] B. Piwowarski, G. Dupret, and R. Jones. Mining user web search activity with layered bayesian networks or how to capture a click in its context. *In The International Conference on Web Search and Data Mining*, WSDM, 2009.

[16] E. Rochd, M. Quafafou, and M. Aznag. Encoding local correspondence in topic models. *In IEEE International Conference on Tools for Artificial Intelligence*, ICTAI, 2013.

[17] J. Schafer, J. Konstan, and J. Riedl. Recommender systems in e-commerce. *In ACM E-Commerce Conference*, 1999.

[18] J. Srivastava, R. Cooley, M. Deshpande, and P. Tan. Web usage mining : discovery and applications of usage patterns from web data. *In ACM SIGKDD Explorations Newsletter*, 2000.

[19] R. Suguna and D. Sharmila. Association rule mining for web recommendation. *In The International Journal on Computer Science and Engineering*, Vol. 4 No. 10, Oct 2012.

[20] G. Xu, Y. Zhang, and X. Yi. Modeling user behavior for web recommendation using lda model. *In The International Conference on Web Intelligence and Intelligent Agent Technology*, 2008.